CONSERVATION AND EFFICIENCY IN LEAST SQUARES FINITE ELEMENT METHODS

BY

JAMES H. LAI

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2012

Urbana, Illinois

Doctoral Committee:

Associate Professor Luke N. Olson, Chair
Dr. Pavel B. Bochev, Sandia National Laboratories
Professor William D. Gropp
Professor Michael T. Heath

Abstract

Two of the main aspects in the numerical solution of partial differential equations include accurate discretizations and efficient solutions of the algebraic equations. With respect to discretizations, conservation is often sought after. However, least-squares finite element methods are known to be not mass conserving when solving fluid flow problems. In this dissertation we develop mass conservative least-squares finite element methods for the Stokes and Navier-Stokes equations through the use of discontinuous finite element spaces. We formulate two divergence free formulations using both a discontinuous stream-function and a locally divergence free basis and we present a thorough numerical study of both methods.

This dissertation is also concerned with the efficient solution of algebraic equations via multigrid methods. Specifically, we formulate multigrid methods for high-order $H(curl)$ conforming finite elements. Such elements are often used in mimetic discretizations of Maxwell's equations often solved in electromagnetic applications. Efficient multigrid methods for high-order H^1 conforming finite elements and also for the lowest-order $H(curl)$ basis have been extensively studied in recent research. We draw upon elements of both algorithms to formulate multigrid methods for high-order $H(curl)$ finite elements for hierarchical and interpolatory type.

To my parents

Acknowledgments

I am grateful for all the friends and family members who have been with me throughout my graduate career.

- My advisor Luke Olson, who introduced me to the beauty of finite elements and multigrid methods, and for his advice and guidance in completing this dissertation.
- Pavel Bochev for the opportunities to work and research at Sandia National Laboratories. Without his insight and dedication to least-squares finite elements this work would not have been possible.
- Michael Heath for always encouraging me to present at the Scientific Computing Seminar and ultimately teaching me how to give great presentations.
- My friends in the Scientific Computing Group – Nana Arizumi, Elena Caraba, Jehanzeb Chaudhry, Steven Dalton, Russ Hewett, Mark Gates, Hormozd Gahvari, Adam Reichert, Mitya Yershov, Michael Wolf, Matt Wrobel – for sharing the graduate school experience with me.
- Anargyros Papageorgiou, my mentor at Columbia University, for sparking my interest in scientific computing and for his advice in life and in graduate school.
- Alex Jiang for reminding me of the joys of college life and for his support and friendship.
- My friends from college – Ronnie Cheng, Dan Gant, Rushan Guan, Chris Ho, and Heikki Virks-Lee.
- My friends from East Brunswick – Kyle Cheng, Justin Cheng, Jon Doan, Mike Liu, Kevin Sun, and Steve Wang.
- Alex Pien for keeping me company through those late nights during the writing of this dissertation.
- John and Wendy, who have been such great role models for me in school and in life.
- Jerry for his support and faith in me and all the hours we spent talking about everything and nothing.
- Christopher for his youthful joy and limitless potential - can't wait to watch you grow up!

- Gus and Rex for their company, their unconditional love, and their furriness.
- My grandma, whose care and love helped raise me from childhood into adulthood.
- My parents for their years of patient guidance and support. Thank you for teaching me the importance of hard work and a good education, encouraging me to purse a graduate degree, and never giving up on me. I would not be where I am today without your love and helping hands.
- Wanyi, who has been with me every step of the way since high school. I can't wait to share the rest of my life with you.

Table of Contents

Chapter 1

Introduction

The efficient solution to partial differential equations (PDEs) is of interest in many areas of science and engineering. Most PDEs do not exhibit analytical solutions and hence must be solved numerically. Two of the main steps in solving numerical PDEs are the discretization of the continuous problem and the solution of the resulting discrete system. Finite elements are a popular choice for discretizing PDEs because of their versatility and their well developed theoretical aspects. Furthermore, discretization by finite elements often results in discrete problems that are sparse and in many cases have desirable properties amenable to efficient iterative methods.

Several diffrent types of finite element methods exist, for example, Galerkin, Petrov-Galerkin, discontinuous-Galerkin, and least-squares methods. In particular, least-squares finite element methods (LSFEM) are a systematic approach to solving PDEs based on an unconstrained optimization principle. Least-squares methods yield symmetric postitive definite (SPD) algebraic systems which are often efficiently solved using iterative methods.

Although there are several advantageous properties of LSFEMs, their use in practice remains limited. Due to the minimization approach, well-posed least-squares methods are always convergent. However, for a given discrete problem the equations of the PDE may not be satisfied exactly. For some PDEs this may lead to inaccurate and unphysical approximatations. For example, in fluid flow governed by the Stokes and Navier-Stokes equations, standard LSFEMs are non conservative and hence result in solutions with mass loss. The goal of this dissertation is to increase the flexibility of least-squares methods by introducing discontinuous elements into the LSFEM framework.

The use of discontinuous elements in the Galerkin setting has grown in popularity for many scientific and engineering probems. Such methods use local element bases and utilize integration by parts to introduce flux terms in the resulting variational form. As a result of the elementwise independence, DG methods may be formulated to be locally conservative [29, 31]. Although the local structure of DG methods form block diagonal matrices, they usually result in saddle point matrices. Due to the indefiniteness of the matrices, preconditioners are more difficult to design. This thesis focuses on discontinuous finite element spaces in

the least-squares setting where the goal is to formulate locally conservative methods while preserving the beneficial aspects of LSFEM.

The second part of this dissertation is concerned with improving the efficiency of iterative solvers for certain problems arising from finite element discretizations. In particular, multigrid has become the method of choice when solving finite element systems. However, optimality is limited to M-matrices which usually arise from H^1-elliptic discretizations. Algebraic multigrid methods have been successful in extending the efficient solution to systems beyond M-matrices.

High-order finite elements are often used in practice due to their excellent convergence properties where highly accurate solutions are possible without extremely refined meshes. However, with the accelerated convergence properties comes reduced sparsity and higher condition numbers in the matrix. In recent years, algebraic multigrid preconditioners for high-order H^1 discretizations has been explored with much success see [52] and the references therein.

In many engineering applications such as electromagnetics mimetic (or compatible) finite elements are popular. Typical mimetic finite elements include curl-conforming or Nédélec elements and divergence-conforming or Raviart-Thomas elements. Such elements are consistent with physical laws, vector calculus identities, and the De Rham complex at the discrete level. In many cases standard H^1 solutions are unphysical where mimetic finite elements eliminate such behavior. Therefore, mimetic finite elements are increasingly used in practice. Although high-order versions of such elements exist, the discrete system is difficult to solve. Efficient solvers and preconditioners for such discretizations remain to be seen. In this dissertation we introduce several efficient multigrid methods that for high-order curl conforming discretizations.

1.1 Dissertation outline

This dissertation is concerned with two broad aspects of solving numerical PDEs: accurate discretizations and efficient solvers. In the remainder of Chapter 1, we define notation used throughout the dissertation. In Chapter 2 we define the least-squares framework and the governing equations of interest. We introduce the Stokes and Navier-Stokes equations from fluid dynamics. We review standard least-squares formulations and demonstrate their numerically properties. In particular, we show the mass conservation properties of such formulations on a series of numerical domains. We introduce two novel methods that resolve the problems in mass conservation while maintaining the advantageous properties of LSFEMs. In Chapter 3 we discuss multigrid methods. We review the basics of multigrid methods and introduce the target problems of our

method – i.e., high-order $H(curl)$ discretizations of the eddy current problem from electromagnetics. We look at two high-order bases for such problems and develop efficient multigrid methods for both bases. In Chapter 4 we discuss implementational aspects of the work in the dissertation. It is meant for reproducibility of the results in the work. We conclude the results of this dissertation in Chapter 5.

1.2 Notation

We denote $\Omega \subset \mathbb{R}^n$ for $n = 2, 3$ a bounded and simply connected region with Lipschitz continuous boundary $\Gamma = \partial\Omega$. $L^2(\Omega)$ is the space of all square integrable functions with norm and inner product denoted $\|\cdot\|_0$ and $(\cdot,\cdot)_0$ respectively while $L^2_0(\Omega)$ denotes the subspace of $L^2(\Omega)$ consisting of functions with zero mean, i.e.,

$$L^2_0(\Omega) = \{u \in L^2(\Omega) \mid \int_\Omega u = 0\}. \tag{1.1}$$

Variational forms of PDEs involve Sobolev spaces. The Sobolev space of order k, is denoted as H^k and is the set of all square integrable functions that have k weak derivatives. We denote the kth order Sobolev norm and inner product by $\|\cdot\|_k$ and $(\cdot,\cdot)_k$ respectively. More formally,

$$H^k(\Omega) = \{u \in L^2(\Omega) \mid \|u\|_k < \infty\}\ , \tag{1.2}$$

where

$$\|u\|_k = \sqrt{(u,u)_k} = \left(\sum_{|\alpha|\le k} \|\partial^\alpha u\|_0^2\right)^{1/2}\ , \tag{1.3}$$

and ∂^α denotes the weak derivative of order α. For $k = 0$, $H^k(\Omega) = L^2(\Omega)$. $H^k_0(\Omega)$ is the closed subspace of $H^k(\Omega)$ of functions that have vanishing trace on Γ. Negative order Sobolev spaces are the duals of their positive order counter parts with the norm

$$\|u\|_{-k} = \sup_{v \in H^k_0(\Omega)} \frac{(u,v)_0}{\|v\|_k}\ . \tag{1.4}$$

In this dissertation we deal with vector valued functions. Vector valued functions and their associated function spaces are denoted in bold. For example, $\boldsymbol{u} = (u_1, u_2)$ is a vector field in $\mathbb{R}^2$ and $\mathbf{H}^1(\Omega)$ is the Sobolev space of vector valued functions in which each component is in $H^1(\Omega)$.

In two dimensions, the curl of a scalar and vector function are given by

$$\nabla \times \omega = \begin{bmatrix} \omega_y \\ -\omega_x \end{bmatrix} , \tag{1.5}$$

$$\nabla \times \boldsymbol{u} = u_{2x} - u_{1y} . \tag{1.6}$$

In two dimensions, the curl of a scalar function ω (1.5) is also known as $\nabla^{\perp}\omega$ and the curl of a vector function $\boldsymbol{u}$ (1.6) is also known as $rot(\boldsymbol{u})$.

We denote a conforming finite element partition of the domain Ω as $\mathcal{K} = \{\kappa\}$ where κ is either a quadrilateral or triangle in two dimensions and a hexahedron or tetrahedron in three dimensions. We denote V^r the C^0 finite element subspace of $H^1(\Omega)$ of degree r. V^r is also known as the nodal finite element space. For example, V^r is commonly taken to be continuous piecewise polynomials of degree r.

The spaces $H(curl, \Omega)$ and $H(div, \Omega)$ denote the space of functions in $\mathbf{L}_2(\Omega)$ whose curls and divergence are square integrable respectively. That is,

$$\begin{aligned} H(curl, \Omega) &= \{\boldsymbol{u} \in \mathbf{L}_2(\Omega) \mid \nabla \times \boldsymbol{u} \in \mathbf{L}_2(\Omega)\} , \\ H(div, \Omega) &= \{\boldsymbol{u} \in \mathbf{L}_2(\Omega) \mid \nabla \cdot \boldsymbol{u} \in L_2(\Omega)\} . \end{aligned} \tag{1.7}$$

Conforming finite element spaces of degree r for $H(curl, \Omega)$ and $H(div, \Omega)$ are denoted as $\mathbf{Q}^r$ and $\mathbf{F}^r$. Elements of $\mathbf{Q}^r$ are commonly known as edge elements, Nédélec elements or Whitney 1-forms. Elements of $\mathbf{Q}^r$ are vector finite elements that enforce tangential continuity across element boundaries. Elements of $\mathbf{F}^r$ are known as face elements or Raviart-Thomas elements and enforce normaly continuity across element interfaces.

Chapter 2

Least-squares finite element methods

Finite element methods solve the variational or weak form of a given partial differential equation (PDE). A well posed weak problem is guaranteed to have a unique solution in a given Hilbert space. Perhaps the most often used are Galerkin methods. Standard Galerkin methods obtain weak forms via the method of weighted residuals (multiplication by test functions and integration by parts). In the case of many variables in different function spaces, mixed Galerkin methods are often used. However, such methods require restrictive conditions on the compatability of the spaces governed by the inf-sup condition. Mixed methods result in saddle point problems which are indefinite and restrict the choice of iterative methods used to solve the discrete system. In this chapter, we focus on least-squares methods where stability, in contrast to standard Galerkin methods, is independent of the approximating spaces and results in symmetric positive definite matrices. Due to the flexibility in choosing finite element spaces and its beneficial computational properties, we are interested in least-squares finite element methods.

We review standard least-squares theory in Section 2.1 and explore standard C^0 methods for the Stokes and Navier-Stokes equations. In particular, we examine the mass conservation of such methods and show in Section 2.3.1 that in some cases, standard methods perform very poorly. A focus of this dissertation aims at improving the mass conservation of least-squares methods in order to further its usage in practice. Our approach is systematic and we allow discontinuous approximating spaces in order to impose elementwise conservation. We propose two methods using stream functions and divergence free bases in Sections 2.5 and 2.6, respectively. This work is also found in [22] and [23]. In Section 2.7 we extend the methods to the full Navier-Stokes equations.

2.1 Least-squares formulation

An alternative to Galerkin methods are least-squares methods which have been well studied in literature. Least-squares finite element methods (LSFEMs) solve PDEs using a minimization approach. The minimization is constructed so that the solution to the optimization problem is the solution to the differential

equation. Let $\mathcal{L} : V \to Q$ where V and Q are Hilbert spaces, and consider the differental equation

$$\mathcal{L}u = f. \tag{2.1}$$

We form a least-squares functional $J(u; f)$ comprised of the residuals of the PDE

$$J(u; f) = \|\mathcal{L}u - f\|_Q^2. \tag{2.2}$$

The least-squares solution is given by minimizing the functional over the given Hilbert space, i.e. *find* $u \in V$ *such that*

$$u = \arg\min_{u \in V} \; J(u; f) \to \tag{2.3}$$

The solution to (2.3) is the zero of the first variational derivative of (2.2).

$$\frac{d}{d\tau} J(u + \tau v; f)|_{\tau=0} = 0. \tag{2.4}$$

Expanding (2.4), we find that

$$\begin{aligned} \frac{d}{d\tau}(\mathcal{L}(u + \tau v) - f, \mathcal{L}(u + \tau v) - f)_Q|_{\tau=0} &= 0 \ , & (2.5)\\ (\mathcal{L}u - f, \mathcal{L}v)_Q &= 0 \ , & (2.6)\\ (\mathcal{L}u, \mathcal{L}v)_Q &= (f, \mathcal{L}v)_Q \ . & (2.7) \end{aligned}$$

Therefore, the solution that minimizes $J(u; f)$ is given by the following weak variational problem: Find $u \in V$ such that

$$(\mathcal{L}u, \mathcal{L}v)_Q = (f, \mathcal{L}v)_Q \tag{2.8}$$

for all $v \in V$.

The bilinear form, $a(\cdot, \cdot) = (\mathcal{L}u, \mathcal{L}v)$, in (2.8) is clearly symmetric. We say that the functional, $J(u; f)$, is norm equivalent on a Hilbert space Q if there exist constants $\alpha, \beta > 0$ such that

$$\|u\|_Q^2 \leq J(u; f) \leq \beta \|u\|_Q^2 \ , \tag{2.9}$$

for all $u \in Q$.

In defining a norm equivalent functional, the resulting weak problem (2.8) is necessarily continuous and coercive. When compared with Galerkin methods, which deal directly with the bilinear form, well-posedness

proofs for LSFEMs are concerned with showing that the defined funtional (2.2) is norm equivalent on the given Hilbert space. Furthermore, we have that the resulting bilinear form that minimizes the least-squares functional is symmetric and positive definite.

To avoid the necessity of higher-order Sobolev spaces, the differential operator $\mathcal{L}$ is usually given in first-order form. The conversion of $\mathcal{L}$ into an equivalent first-order form requires the introduction of new variables and in many cases becomes a system of differential equations. In this respect, least-squares methods are also known as first-order sytem of least-squares (FOSLS) methods.

2.2 Governing equations

One of the most widely solved PDEs in engineering is the Navier-Stokes equations. The Stokes and Navier-Stokes equations model incompressible fluid flow and are solve in many aspects of engineering. This necessitates the need for accurate discretizations of the equations. The Navier-Stokes equations are given by

$$\begin{cases} -\triangle \boldsymbol{u} + Re(\boldsymbol{u} \cdot \nabla)\boldsymbol{u} + \nabla p = \boldsymbol{f} & \text{on } \Omega \\ \nabla \cdot \boldsymbol{u} = 0 & \text{on } \Omega \end{cases} \tag{2.10}$$

Where $\boldsymbol{u}$ denotes the velocity, p denotes the pressure and $\boldsymbol{f}$ denotes the body force. $Re > 0$ is the Reynolds number which controls the importance of the nonlinear convective term. The system is closed by the velocity boundary condition

$$\boldsymbol{u} = 0 \quad \text{on } \partial\Omega \tag{2.11}$$

and the zero mean pressure condition

$$\int_{\Omega} p = 0 \tag{2.12}$$

The system of PDEs in (2.10) is a nonlinear system of differential equations. We therefore first look at the Stokes equations as methods and theory developed for the Stokes equations motivate the approach to the Navier-Stokes equations. The Stokes equations drop the nonlinear convective term $Re((\boldsymbol{u}\cdot)\nabla\boldsymbol{u})$ in the first equation of (2.82) and are given by

$$\begin{cases} -\triangle \boldsymbol{u} + \nabla p = \boldsymbol{f} & \text{on } \Omega \\ \nabla \cdot \boldsymbol{u} = 0 & \text{on } \Omega \end{cases} \tag{2.13}$$

with velocity boundary conditions (2.11) and zero mean pressure constraint (2.12).

2.2.1 First-order formulations

As mentioned in Section 2.1 the differential equations need to be written in an equivalent first order form. There are many different ways to introduce new variables in order to write (2.13) in first order form, we introduce a new variable

$$\omega = \nabla \times \boldsymbol{u} \ , \tag{2.14}$$

which we call the vorticity. This is a useful formulation as the vorticity is commonly sought after in numerical simulations of fluid flow. Using the vector calculus identity

$$\nabla \times \nabla \times \boldsymbol{u} = -\triangle \boldsymbol{u} + \nabla(\nabla \cdot \boldsymbol{u}) \ , \tag{2.15}$$

the definition of vorticity (2.14), and the fact $\nabla \cdot \boldsymbol{u} = 0$, the velocity is replaced by the vorticity in the first equation of (2.13). The velocity-vorticity-pressure (VVP) formulation of the Stokes equations is given by

$$\begin{cases} \nabla \times \omega + \nabla p = \boldsymbol{f} & \text{on } \Omega \ , \\ \nabla \times \boldsymbol{u} - \omega = 0 & \text{on } \Omega \ , \\ \nabla \cdot \boldsymbol{u} = 0 & \text{on } \Omega \ . \end{cases} \tag{2.16}$$

Other commonly used first order formulations include the velocity-stress-pressure, and velocity-gradient-velocity-pressure where the stress tensor and gradient of velocity are introduced as new variables. The VVP formulation (2.16) remains widely used, we therefore consider LSFEMs for (2.16).

Once the PDE is written in first order form, we formulate least-squares functionals in which the minimizer coincides with the solution to the PDE.

2.3 Continuous LSFEM

Continuous formulations for least-squares finite element methods have been extensively studied in literature [4, 17, 18, 43]. For the Stokes and Navier-Stokes equations, there is well developed theory behind the well posedness of such formulations [10, 13, 16]. The key to a well-posed LSFEM is a norm-equivalent least-squares functional [18]. For the VVP Stokes system, we have the *a priori* bound

$$\|\boldsymbol{u}\|_1 + \|\omega\|_0 + \|p\|_0 \leq C \left(\|\nabla \times \omega + \nabla p\|_{-1} + \|\omega - \nabla \times \boldsymbol{u}\|_0 + \|\nabla \cdot \boldsymbol{u}\|_0 \right) \tag{2.17}$$

for any $\boldsymbol{u} \in \mathbf{H}_0^1(\Omega) = [H_0^1(\Omega)]^d$, $\omega \in L^2(\Omega)$, and $p \in L_0^2(\Omega)$. This bound implies that the *negative norm* functional

$$J_{-1}(\boldsymbol{u}, \omega, p; \boldsymbol{f}) = \|\nabla \times \omega + \nabla p - \boldsymbol{f}\|_{-1}^2 + \|\nabla \times \boldsymbol{u} - \omega\|_0^2 + \|\nabla \cdot \boldsymbol{u}\|_0^2 \tag{2.18}$$

is norm equivalent on $X = \mathbf{H}_0^1(\Omega) \times L^2(\Omega) \times L_0^2(\Omega)$, and that the least-squares principle: *find* $(\boldsymbol{u}, \omega, p) \in X$ *such that*

$$J_{-1}(\boldsymbol{u}, \omega, p; \boldsymbol{f}) \leq J_{-1}(\boldsymbol{v}, \xi, q; \boldsymbol{f}) \quad \forall (\boldsymbol{v}, \xi, q) \in X \tag{2.19}$$

is a well-posed unconstrained minimization problem whose minimizer coincides with the solution of the VVP Stokes system.

Formally, a well-posed LSFEM is derived by restricting the minimization in (2.19) to a finite element subspace $X^h \subset X$. However, this method is impractical because

$$\|u\|_{-1}^2 = \|(-\triangle)^{-1/2} u\|_0^2 \,. \tag{2.20}$$

That is, computation of the negative norm requires inversion of the Laplace operator [24]. In order to obtain a practical method the negative norm in (2.18) must be replaced by a computable discrete approximation. The diagonal operator

$$(-\triangle)^{-1/2} \mapsto h\mathcal{I} \,, \tag{2.21}$$

where $\mathcal{I}$ is the identity, provides a simple, yet sufficiently accurate approximation of the negative norm [18]. The discrete negative norm can be approximated by

$$\|\phi^h\|_{-h}^2 \approx h^2 \|\phi^h\|_0^2 \tag{2.22}$$

Using (2.22) we obtain a discrete version of (2.18)

$$J_{-1}^h(\boldsymbol{u}^h, \omega^h, p^h; \boldsymbol{f}) = h^2 \|\nabla \times \omega^h + \nabla p^h - \boldsymbol{f}\|_0^2 + \|\nabla \times \boldsymbol{u}^h - \omega^h\|_0^2 + \|\nabla \cdot \boldsymbol{u}^h\|_0^2 \tag{2.23}$$

and the following discrete least-squares principle: *find* $(\boldsymbol{u}^h, \omega^h, p^h) \in X_r^h$ *such that*

$$J_{-1}^h(\boldsymbol{u}^h, \omega^h, p^h; \boldsymbol{f}) \leq J_{-1}^h(\boldsymbol{v}^h, \xi^h, q^h; \boldsymbol{f}) \qquad \forall (\boldsymbol{v}^h, \xi^h, q^h) \in X_r^h \tag{2.24}$$

where

$$X_r^h = \mathbf{V}^r \cap \mathbf{H}_0^1(\Omega) \times V^{r-1} \times V^{r-1} \cap L_0^2(\Omega) \quad r > 1 \,. \tag{2.25}$$

We refer to the method (2.23)–(2.24) as the *weighted* L^2 LSFEM. This method is a well-posed and optimally convergent formulation [14]. The *minimal approximation condition* $r > 1$ is required for optimal convergence rates. The violation by using, for example, V^1 elements for all variables in (2.25), reduces the accuracy of the least-squares solution; see [14, 15].

The following result holds [18, Theorem 7.14, p.262].

Theorem 2.3.1. *Let $(\boldsymbol{u}^h, \omega^h, p^h) \in X_r^h$, $r > 1$ be a solution to (2.23). Assume that the exact solution of the VVP Stokes system (2.16) is such that $\boldsymbol{u} \in \mathbf{H}^{r+1}(\Omega)$, $\omega \in H^r(\Omega)$ and $p \in H^r(\Omega)$. There exists a constant $C > 0$ such that*

$$\begin{aligned} \|\boldsymbol{u} - \boldsymbol{u}^h\|_1^2 + \|\omega - \omega^h\|_0 + \|p - p^h\|_0 &\leq Ch^r \left(\|\boldsymbol{u}\|_{r+1} + \|\omega\|_r + \|p\|_r\right) \\ \|\omega - \omega^h\|_1 + \|p - p^h\|_1 &\leq Ch^{r-1} \left(\|\boldsymbol{u}\|_{r+1} + \|\omega\|_r + \|p\|_r\right) . \end{aligned} \tag{2.26}$$

Theorem 2.3.1 shows that approximating the discrete negative norm with a weighted L^2-norm (2.22) results in an optimally convergent method. Therefore, in our studies, we restrict attention to implementations of (2.24) using the *equal-order* space

$$\overline{X}_r^h = \mathbf{V}^r \cap \mathbf{H}_0^1(\Omega) \times V^r(\Omega) \times \breve{V}^r(\Omega) \quad r > 1 \,, \tag{2.27}$$

where $\breve{V}^r$ is the pressure space constrained at a single node on the boundary. For simplicity, we use this approach instead of enforcing (2.12). Because the term involving the pressure variable is essentially a Laplacian, the nullspace for the pressure variable is one-dimensional and consists of constant functions. The two approaches to elimination of the one-dimensional null-space in the discrete system are equivalent. However, the choice affects the convergence of the iterative method used to solve the system. As compared with implementing (2.12), setting the pressure space at a single node, increases the condition number. A comparison and implementation details can be found in [18, Section 7.6.4].

A least-squares method with improved norm equivalence is obtained by using the operator

$$(-\triangle)^{-1/2} \mapsto h\mathcal{I} + (\mathcal{L}^h)^{1/2} \,, \tag{2.28}$$

where $\mathcal{L}^h$ is a spectrally equivalent preconditioner for the Laplace operator [24]. This operator results in a *discrete negative norm*

$$\|\phi^h\|_{-h}^2 = h^2 \|\phi^h\|_0^2 + \|(\mathcal{L}^h)^{1/2} \phi^h\|_0^2 \,, \tag{2.29}$$

which, for finite element functions, is equivalent to the negative norm (1.4). Using (2.29) we obtain a discrete negative norm version of (2.18), namely

$$J_{-h}(\boldsymbol{u}^h, \omega^h, p^h; \boldsymbol{f}) = \|\nabla \times \omega^h + \nabla p^h - \boldsymbol{f}\|_{-h}^2 + \|\nabla \times \boldsymbol{u}^h - \omega^h\|_0^2 + \|\nabla \cdot \boldsymbol{u}^h\|_0^2 \ . \tag{2.30}$$

This is a well-posed least-squares formulation, which we term the *discrete negative norm* LSFEM. The latter has no discretization limitations so using the equal-order space (2.27) with $r = 1$ suffices [11]. The error estimates in Theorem 2.3.1 continue to hold for the new method, including the case when $r = 1$ in (2.27).

The least-squares functionals (2.23) and (2.30) differ only by their treatment of the momentum equation. The use of the discrete negative norm (2.29) to measure the residual leads to improved conditioning [24] in linear systems, resulting in more efficient solution by preconditioned conjugate gradients. However, compared to the weighted L^2 LSFEM, implementation of the discrete negative norm LSFEM is more involved [11]. Moreover, the condition number growth with respect to mesh refinement is $O(h^{-4})$ for (2.23) while for (2.30) the growth is $O(h^{-2})$, similar to a Galerkin approach.

Because we are interested in the mass conservation of LSFEMs, the methods do not depend on the treatment of the momentum equation. Its application to both the weighted L^2 and the discrete negative norm LSFEMs follows a similar process. Therefore, for simplicity and in order to focus on the issue of mass conservation, we use the simpler setting of (2.23) to motivate the approach and discuss the implementation of the resulting, locally conservative LSFEMs.

We now quantify the problem of mass loss in LSFEMs for two standard test problems. We restrict attention to the weighted L^2 LSFEM; the situation with (2.30) for these examples follows similarly. Theorem 2.3.1 asserts that both (2.23) and (2.30) are optimally accurate for all sufficiently smooth exact solutions of the Stokes equations. This implies that asymptotically $\|\nabla \cdot \boldsymbol{u}\| \to 0$, as $h \to 0$. However, on a given fixed mesh size this term is not necessarily small and convergence is not guaranteed for solutions with reduced smoothness. The following examples support these concerns and highlight that (2.23) experiences significant mass loss in certain settings.

To this end we consider two standard test problems: the backward-facing step flow, shown in Figure 2.1, and a channel flow past a cylinder, shown in Figure 2.2. For the backward-facing step the domain is the rectangle $[0, 10] \times [0, 1]$ with a reentrant corner at $(2, 0.5)$. The velocity boundary condition is specified as follows. On the inflow ($x = 0$) and outflow ($x = 10$) walls

$$\boldsymbol{u}_{in} = \begin{bmatrix} 8(y - 0.5)(1 - y) \\ 0 \end{bmatrix} \quad \text{and} \quad \boldsymbol{u}_{out} = \begin{bmatrix} y(1 - y) \\ 0 \end{bmatrix} , \tag{2.31}$$

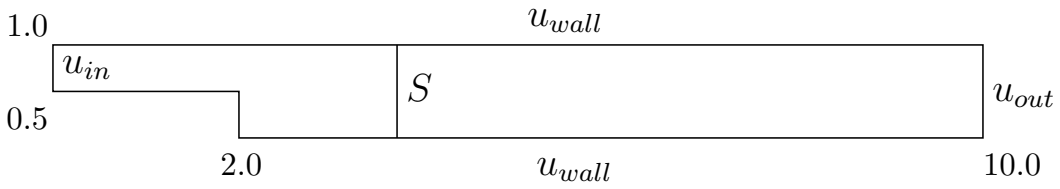


Figure 2.1: *Geometry of the first test problem: backward-facing step.*

respectively. Along all other parts of the boundary $\boldsymbol{u}_{wall} = \mathbf{0}$ is enforced. For this domain we use a mesh of 900 rectangular elements.

The geometry of the second test problem is given by the rectangle $[-1, 3] \times [-1, 1]$ with a disk of radius $r > 0$ centered at $(0, 0)$, removed from the domain. We consider two cases: $r = 0.6$ and $r = 0.9$. The velocity boundary condition for this problem is set as follows. On the inflow ($x = -1$), outflow ($x = 3$), top ($y = 1$), and bottom ($y = -1$) walls

$$\boldsymbol{u}_{in} = \boldsymbol{u}_{out} = \boldsymbol{u}_{wall} = \begin{bmatrix} (1-y)(1+y) \\ 0 \end{bmatrix}, \tag{2.32}$$

and on the surface of the "cylinder" $\boldsymbol{u}_{cyl} = \mathbf{0}$. Therefore, velocity is set to zero on all parts of the boundary except for the inflow and the outflow portions of $\partial\Omega$. The mesh for this problem comprises of 1296 triangles when $r = 0.6$ and 1104 triangles when $r = 0.9$.

The velocity boundary condition in both problems is compatible with $\nabla \cdot \boldsymbol{u} = 0$ because fluid enters and leaves the domain only through the inflow and the outflow boundaries, respectively and

$$\int_{\Gamma_{in}} \boldsymbol{u}_{in} \cdot \boldsymbol{n} \, d\ell = \int_{\Gamma_{out}} \boldsymbol{u}_{out} \cdot \boldsymbol{n} \, d\ell \, .$$

As a result, to assess the mass conservation in the least-squares solution, we measure the total mass flow across a sequence of vertical surfaces connecting the top and the bottom sides of the computational domain. The lines marked by "S" in Figures 2.1-2.2 show two typical examples of such surfaces for the two test problems. Because the greatest mass loss for the backward-facing step is expected near the reentrant corner we always place one of the surfaces at $x = 2$. For the second test problem we always measure the flow across the surface at $x = 0$ where the domain narrows due to the cylindrical cutout.

In both test problems, velocity is zero on all parts of the boundary except Γ_{in} and Γ_{out}. It follows from the divergence theorem that

$$\int_{\Gamma_{in}} \boldsymbol{u} \cdot \boldsymbol{n}_{in} \, d\ell = \int_{S} \boldsymbol{u} \cdot \boldsymbol{n}_{S} \, d\ell \, ,$$

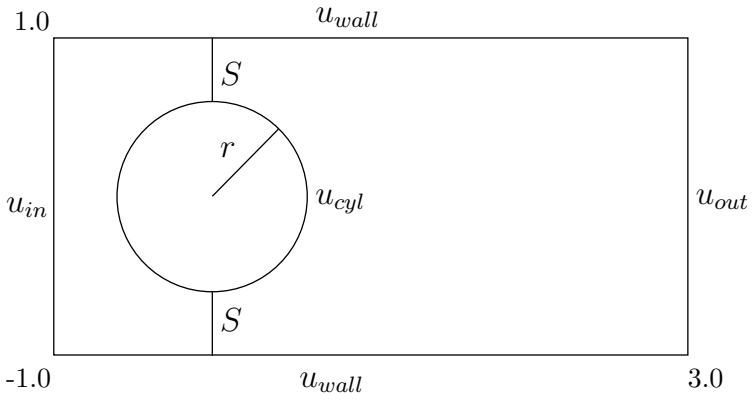


Figure 2.2: *Geometry of the second test problem: flow past a cylinder.*

for any S connecting the top and bottom walls of the domain. Therefore, mass conservation is quantified by the percent mass loss across the surface S, defined as follows:

$$\%m_{loss} = \frac{\displaystyle\int_{\Gamma_{in}} \boldsymbol{u}\cdot\boldsymbol{n}_{in}\,d\ell - \int_{S} \boldsymbol{u}\cdot\boldsymbol{n}_{S}\,d\ell}{\displaystyle\int_{\Gamma_{in}} \boldsymbol{u}\cdot\boldsymbol{n}_{in}\,d\ell} \times 100\,. \tag{2.33}$$

2.3.1 Mass conservation for C^0 methods

To assess mass conservation properties of the weighted L^2 LSFEM we solve the two test problems using the following modified version of the least-squares functional (2.23)

$$J^h_\mu(\boldsymbol{u}^h,\omega^h,p^h;\boldsymbol{f}^h) = h^2\|\nabla\times\omega^h + \nabla p^h - \boldsymbol{f}^h\|_0^2 + \|\nabla\times\boldsymbol{u}^h - \omega^h\|_0^2 + \mu\|\nabla\cdot\boldsymbol{u}^h\|_0^2 \tag{2.34}$$

and the equal order C^0 space (2.27) with $r = 2$. This modification was a previously proposed way to improve mass conservation in least-squares methods [33]. By increasing μ we increase the relative importance of the residual of the continuity equation, thereby promoting mass conservation. The additional weight on the divergence free constraint for the velocity is reminiscent of penalty methods from optimization. As we increase the penalty parameter, the equation is enforced more strongly. However, increasing the penalty parameter causes imbalance in the terms and hence changes the norm equivalence of the functional. In addition, as the penalty parameter is increased, the condition number of the matrix increases. In our study we use $\mu = 1$, $\mu = 10$ and $\mu = 20$.

Our results are summarized in Figure 2.3. We see that for $\mu = 1$ the least-squares solution of the backward-facing step problem exhibits severe mass loss in excess of 50% of the total mass near the reentrant

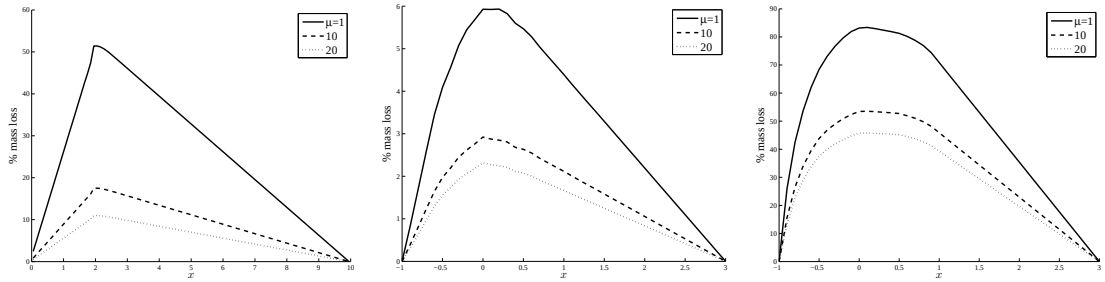


Figure 2.3: *Percent mass loss of (2.34) for the backward-facing step (left panel) and the flow past a cylinder with $r = 0.6$ (center panel) and $r = 0.9$ (right panel). Values are computed using (2.33) along vertical lines placed at every 0.1 units along the x-axis. A total of 100 lines are used for the backward-facing step and 40 lines are used for the flow past a cylinder.*

corner. Increasing μ does improve conservation, however, mass loss remains unacceptably high even for $\mu = 20$. The mass loss in the second test problem with $r = 0.6$ is less severe but still noticeable at 6%. In this case, setting $\mu = 20$ reduces the loss of mass across the narrowest part of the domain to about 2%. However, as the radius increases to $r = 0.9$, the mass loss at this location jumps to over 80%. Moreover, setting $\mu = 20$ does not yield noticeable improvement and mass loss remains unacceptably high in excess of 40%. These examples are indicative of the inherent problems with mass conservation in conventional LSFEMs.

We note that a significant increase of μ is not recommended as it also reduces the accuracy of the other terms in the functional, thus compromising other qualities such as conservation of momentum. Indeed, by increasing the weight of a single term in the least-squares functional, it is *decreasing* the importance of the other terms. Thus, by choosing a large weight for μ to promote mass conservation, we are effectively demoting conservation of momentum.

Exact element-wise mass conservation with C^0 elements has been achieved in the so-called *restricted* least-squares method [28]. In the restricted LSFEM, mass conservation on each element is added as an explicit constraint leading to the following constrained minimization problem:

$$\min_{X_r^h} J_{-1}^h(\boldsymbol{u}^h, \omega^h, p^h; \boldsymbol{f}) \quad \text{subject to} \int_K \left(\nabla \cdot \boldsymbol{u}^h\right) dK = 0, \quad \forall K \in \mathcal{K}\,. \tag{2.35}$$

Although (2.35) returns a solution with exact element-wise mass conservation, this constrained optimization problem is typically solved using Lagrange multipliers and results in a saddle-point system which negates the advantages of using least-squares. The constrained optimization problem can also be solved by a penalty approach, which ultimately leads to a formulation similar to (2.34) with large μ. Because the penalty must be large in order to enforce the constraint accurately, the penalty formulation of (2.35) suffers from the same

disadvantages as (2.34) with $\mu \gg 1$.

In the next section we present an alternative approach to improve mass conservation in least-squares methods based on allowing discontinuous velocity spaces in the formulation. Discontinuous spaces allows for the imposition of constraints at the local level as opposed to restrictive global constraints on the entire space.

2.4 Discontinuous LSFEM

Discontinuous finite elements have been used widely in the Galerkin framework where element local basis functions are defined and integration by parts leads to the variational formulation involving fluxes. The discontinuous Galerkin methods result in mixed variational problems. In this section we look at extending the methods from discontinuous Galerkin to least-squares methods. Using standard least-squares principles and ideas from discontinuous Galerkin methods, we will first introduce least-squares functionals over discontinuous spaces. We then show that such formulations do not strongly impose mass conservation on the interiors of the elements. Therefore, the solution is no better than the standard C^0 formulations. We improve upon the discontinuous formulations by strongly enforcing elementwise conservation. In Section 2.5, we use a discontinuous stream function to approximate the velocity space arriving at the discontinuous stream function-vorticity-pressure (dS-VP)formulation. A number of disadvantages arises in choosing the dS-VP formulation and we improve them with a discontinuous velocity-vorticity-pressure formulation with a divergence free basis in Section 2.6. Discontinuous methods often result in increased condition numbers for the matrix, we therefore, introduce a preconditioner for our methods which reduces the condition number growth to the same level as a continuous Galerkin method.

Numerical results in the previous section show that C^0 LSFEMs suffer from mass loss, exceeding 80% of the total mass for some formulations. Furthermore, the remedies available to counter this loss are not satisfactory: weighting strongly the continuity equation residual as in (2.34) reduces conservation of momentum, while using the restricted formulation (2.35) leads to a saddle-point problem.

The usage of mimetic finite element formulations has been explored in [19]. These methods use divergence-conforming elements for the velocity space, such as the Raviart-Thomas element [55], and curl-conforming elements for the vorticity space such as Nédélec elements [50], to achieve exact mass conservation. However, the resulting mimetic LSFEM requires non-standard boundary conditions for the Stokes equations where normal velocity and tangential vorticity boundary conditions are imposed. It is not clear how the method can be adapted for the more practical case of velocity boundary conditions.

Consequently, in order to improve mass conservation in LSFEMs for the Stokes equations with the velocity boundary condition we propose to employ a *discontinuous* finite element approximation of the velocity, while retaining C^0 elements for the rest of the variables. In so doing we achieve two objectives. First, we keep the growth of the degrees of freedom to a minimum, compared to a fully discontinuous formulation. Second, relaxation of the interelement continuity of the velocity space allows for a greater flexibility in the choice of the local finite element approximation of that variable. In particular, it becomes possible to consider locally divergence-free spaces which would have been impractical and restrictive if the global velocity space also had to be H^1-conforming. Raviart-Thomas-like elements which are also H^1-conforming, i.e., continuous across element interfaces, are constructed on rectangular grids using tensor products of one-dimensional quadratic Lagrange and cubic Hermite shape functions [3]. Using such elements yields a least-squares formulation which computes solenoidal velocity fields. However, the scope of such a formulation would be limited to regions that could be meshed entirely by rectangular elements. Our second test problem is one example where this is not feasible. Following these ideas we develop locally mass-conservative least-squares formulations based on functionals (2.23) and (2.30).

In our first functional, we introduce a discontinuous velocity functional where the continuity on the velocity space is relaxed. We show in Section 2.4.1 that although the velocity space is discontinuous, mass conservation is not improved due to loss of mass on the interior of elements. However, the discontinuous velocity space allows for the use of solenoidal fields. An immediate solenoidal field is taken to be the curl of a local stream function. Using such spaces for the velocity gives rise to the discontinuous stream function-vorticity-pressure (dS-VP) formulation described in Section 2.5. The dS-VP exhibits greatly improved mass conservation, however, implicit definition of the velocity field in terms of the stream function creates a number of drawbacks including non-conventional imposition of the velocity boundary condition and increased growth in condition number with mesh refinement. We address these issues by using a piecewise divergence free basis detailed in Section 2.6. We now test the improvement in mass conservation form the dS-VP formulation (2.41) through several computational examples.

2.4.1 Discontinuous velocity least-squares formulation

In this section we relax the continuity requirement for the velocity approximating space. Specifically, we change the approximating space from (2.27) to a space where the first component is discontinuous:

$$\widetilde{X}_r^h = [\mathbf{V}^r] \times V^r \times \breve{V}^r \,. \tag{2.36}$$

To address the discontinuity in the velocity space we modify (2.23) and (2.30). First, the last two terms in these functionals are split into sums over individual elements. Second, we enforce H^1-conformity weakly by adding residuals of the tangential and normal jumps of the velocity across the element interfaces (edges) $\mathcal{E}(\Omega)$; this methodology is consistent with standard least-squares approaches. As a result, at the first stage we are led to the following *discontinuous velocity* versions of (2.23) and (2.30):

$$\begin{aligned}
&\widetilde{J}^h_{-1}(\boldsymbol{u}^h,\omega^h,p^h;\boldsymbol{f}^h) = \\
&\quad h^2\|\nabla\times\omega^h+\nabla p^h-\boldsymbol{f}^h\|_0^2+\sum_{K\in\mathcal{K}}\left(\|\nabla\times\boldsymbol{u}^h-\omega^h\|_{0,K}^2+\|\nabla\cdot\boldsymbol{u}^h\|_{0,K}^2\right) \\
&\qquad +\sum_{e_i\in\mathcal{E}(\Omega)}h^{-1}\left(\alpha_1\|[\boldsymbol{u}\cdot\boldsymbol{n}_i]\|_{0,e_i}^2+\alpha_2\|[\boldsymbol{u}\times\boldsymbol{n}_i]\|_{0,e_i}^2\right)
\end{aligned} \tag{2.37}$$

$$\begin{aligned}
&\widetilde{J}_{-h}(\boldsymbol{u}^h,\omega^h,p^h;\boldsymbol{f}^h) = \\
&\quad \|\nabla\times\omega^h+\nabla p^h-\boldsymbol{f}^h\|_{-h}^2+\sum_{K\in\mathcal{K}}\left(\|\nabla\times\boldsymbol{u}^h-\omega^h\|_{0,K}^2+\|\nabla\cdot\boldsymbol{u}^h\|_{0,K}^2\right) \\
&\qquad +\sum_{e_i\in\mathcal{E}(\Omega)}h^{-1}\left(\alpha_1\|[\boldsymbol{u}\cdot\boldsymbol{n}_i]\|_{0,e_i}^2+\alpha_2\|[\boldsymbol{u}\times\boldsymbol{n}_i]\|_{0,e_i}^2\right)
\end{aligned} \tag{2.38}$$

where $\alpha_1,\alpha_2>0$ control the relative importance of normal and tangential continuity.

The weights of interface residuals are determined through conditions on the trace. Because the trace of an $H^1(\Omega)$ function is well-defined in $H^{1/2}(S)$, where S is surface contained in the closure of Ω, the proper forms of the interface jump residuals are given by

$$\|[\boldsymbol{u}\cdot\boldsymbol{n}_i]\|_{1/2,e_i}^2 \quad\text{and}\quad \|[\boldsymbol{u}\times\boldsymbol{n}_i]\|_{1/2,e_i}^2\,, \tag{2.39}$$

respectively. However, similar to the negative norm formulation, the trace norm is not easily computable, necessitating a more practical alternative. One straightforward approach is to consider the inverse inequality

$$\|\phi^h\|_{1/2,\partial K}^2\le Ch^{-1}\|\phi^h\|_{0,\partial K}^2\,,$$

which holds for most reasonable finite element partitions and suggests the weighted[1] trace norms used in (2.37) and (2.38). Also, as in the case of the discrete negative norm (2.29), there are more sophisticated alternatives to weighted trace norms defined by using special boundary functionals [48].

The functionals (2.37)-(2.38) may also be viewed as extensions of the least-squares formulation for transmission problems [27] to the VVP Stokes system with one important distinction. Namely, the interface coupling terms are applied only to the velocity because the vorticity and the pressure remain approximated

[1]We note that identically weighted norms were used in [40] to weakly enforce the velocity boundary condition.

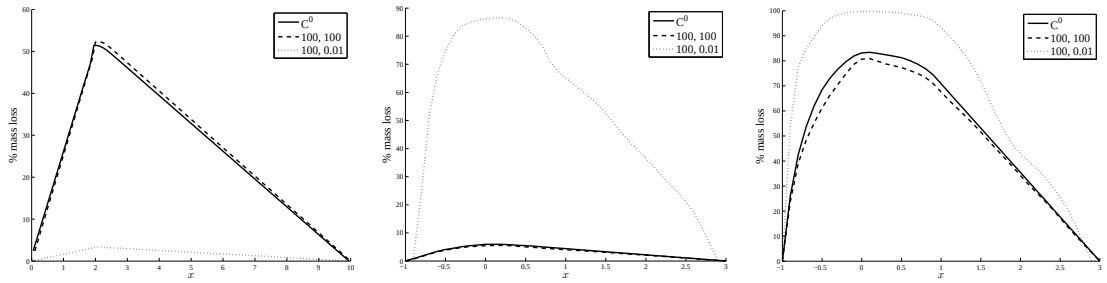


Figure 2.4: *Percent mass loss in the discontinuous velocity least-squares method (2.37) for the backward-facing step (left panel) and the flow past a cylinder with* $r = 0.6$ *(center panel) and* $r = 0.9$ *(right panel). Dashed line corresponds to* $\alpha_1 = \alpha_2 = 100$*, dotted line corresponds to* $\alpha_1 = 100$*,* $\alpha_2 = 0.01$ *and the solid line gives the reference mass loss by the prototype* C^0 *least-squares method (2.23). The legend values are read as* α_1, α_2 *with (2.23) as reference labeled* (C_0)*. Values are computed using (2.33) along vertical lines placed at every 0.1 units along the* x*-axis. A total of 100 lines are used for the backward-facing step and 40 lines are used for the flow past a cylinder.*

by C^0 elements. See [8, 12, 54] for further examples of domain-decomposition and discontinuous least-squares formulations.

In view of divergence-conforming elements, one approach is to improve the mass conservation in the finite element solution of (2.37) and (2.38) by strengthening the normal continuity of the velocity field. On the other hand, the discontinuous velocity formulations (2.37) and (2.38) with $\alpha_1 \gg \alpha_2$ directly target reduction in mass loss for our two test problems. To test this hypothesis we implement (2.37) using the equal-order, discontinuous velocity finite element space (2.36) with $r = 2$ and solve the two test problems with two different choices for α_1 and α_2. The first choice is to set $\alpha_1 = \alpha_2 = 100$, in which case we expect[2] to see mass losses comparable to that in the original C^0 formulation (2.23). The second set of weights $\alpha_1 = 100$, $\alpha_2 = 0.01$ emphasizes normal over tangential continuity. The expectation is that this set of weights leads to an improved mass conservation. Unfortunately, the results shown in Figure 2.4 do not support our conjecture that mass loss can be controlled by interelement continuity alone. Indeed, the left panel in the figure shows that for the backward-facing step problem the second weight combination leads to a *significant improvement* in the mass conservation by reducing the mass loss from over 50% to just over 3%. However, the situation is reversed for the second test problem with $r = 0.6$. Now the choice $\alpha_1 = 100$, $\alpha_2 = 0.01$ leads to a *significant deterioration* of the mass conservation and increases mass loss from 6% in the C^0 formulation to *nearly* 90% in the discontinuous velocity LSFEM. When the radius increases to $r = 0.9$ the same weight combination leads to a nearly *complete* mass loss, as shown in the right panel of Figure 2.4.

These results indicate that the discontinuous velocity formulation (2.37) is not reliable in improving mass conservation with the same choice of weights. Or, more precisely, mass conservation properties are problem

[2]This is because in the limit as $\alpha_1 \to \infty$ and $\alpha_2 \to \infty$, (2.37) recovers the C^0 solution of the weighted L^2 LSFEM method.

dependent. We address these issues by enforcing local conservation on the interior of the elements.

2.5 Stream function-vorticity-pressure

While the discontinuous velocity functionals (2.37) and (2.38) enable some improvements in mass conservation, they do not enforce mass conservation locally on each element. Considering that the velocity space is not subject to any interelement continuity, with (2.37) and (2.38) we have greater flexibility for choosing the velocity representation on each element than with (2.23) and (2.30). In this section we formulate a locally conservative LSFEM for the Stokes equations using a discrete velocity field that is pointwise divergence free on each element. We begin with the discontinuous velocity formulation from Section 2.4.1 and define the velocity field on each element using a local stream function. From vector calculus, it is clear that $\nabla \cdot (\nabla \times \omega) = 0$ for any differentiable ω. Therefore, using this fact, $\nabla \times \omega$ defines a divergence free basis. We set the velocity space to be the following

$$\boldsymbol{u}^h|_K = \nabla \times \psi^h|_K \quad \forall K \in \mathcal{K}\,, \tag{2.40}$$

where $\psi^h \in [V_{r+1}]$ is a discontinuous *stream function*. The finite element space for ψ^h is of one degree higher than the original velocity finite element space to ensure that $\nabla \times \psi^h \in [\mathbf{V}_r]$.

We replace the velocity approximation in (2.37) and (2.38) with the field defined in (2.40). Note that in this definition of $\boldsymbol{u}^h$, we have that $\nabla \cdot \boldsymbol{u}^h = 0$ is automatically satisfied. In response, we drop the residual of the continuity equation from the least-squares functional and add a term that penalizes the jump of the stream function. Furthermore, because velocity is eliminated, the velocity boundary condition is imposed through the stream function. Since $\mathbf{n} \cdot \nabla \times \psi^h$ involves only tangential derivatives of ψ^h, a Dirichlet boundary condition on the stream-function prescribes the normal component of the velocity. We specify the tangential component of the velocity weakly by adding its residual to the least-squares functional. In summary, at the end of the second stage, the discontinuous velocity functionals (2.37) and (2.38) are transformed into the

following *discontinuous stream-function-vorticity-pressure* functionals:

$$\begin{aligned}
&\widetilde{J}^h_{-1}(\psi^h,\omega^h,p^h;\boldsymbol{f}^h) = \\
&\quad h^2\|\nabla\times\omega^h+\nabla p^h-\boldsymbol{f}^h\|_0^2+\sum_{K\in\mathcal{K}}\|\nabla\times\nabla\times\psi^h-\omega^h\|_{0,K}^2 \\
&\qquad +\sum_{e_i\in\mathcal{E}(\Omega)} h^{-1}\left(\alpha_1\|[(\nabla\times\psi^h)\cdot\boldsymbol{n}_i]\|_{0,e_i}^2+\alpha_2\|[(\nabla\times\psi^h)\times\boldsymbol{n}_i]\|_{0,e_i}^2\right) \\
&\qquad +\sum_{e_i\in\mathcal{E}(\Gamma)} h^{-1}\|(\nabla\times\psi^h)\times\boldsymbol{n}_i\|_{0,e_i}^2+\sum_{e_i\in\mathcal{E}(\Omega)} h^{-3}\|[\psi^h]\|_{e_i}^2
\end{aligned} \tag{2.41}$$

$$\begin{aligned}
&\widetilde{J}_{-h}(\psi^h,\omega^h,p^h;\boldsymbol{f}^h) = \\
&\quad \|\nabla\times\omega^h+\nabla p^h-\boldsymbol{f}^h\|_{-h}^2+\sum_{K\in\mathcal{K}}\|\nabla\times\nabla\times\psi^h-\omega^h\|_{0,K}^2 \\
&\qquad +\sum_{e_i\in\mathcal{E}(\Omega)} h^{-1}\left(\alpha_1\|[(\nabla\times\psi^h)\cdot\boldsymbol{n}_i]\|_{0,e_i}^2+\alpha_2\|[(\nabla\times\psi^h)\times\boldsymbol{n}_i]\|_{0,e_i}^2\right) \\
&\qquad +\sum_{e_i\in\mathcal{E}(\Gamma)} h^{-1}\|(\nabla\times\psi^h)\times\boldsymbol{n}_i\|_{0,e_i}^2+\sum_{e_i\in\mathcal{E}(\Omega)} h^{-3}\|[\psi^h]\|_{e_i}^2\,.
\end{aligned} \tag{2.42}$$

The weight for last term in (2.41) and (2.42) is determined by an inverse inequality, analogous to that of the velocity jump terms, but assuming that $\psi\in H^2(\Omega)$ and hence its trace is in $H^{3/2}(S)$ where Ω and S are as defined previously for the discontinuous velocity formulation. The jump of the stream-function is necessary for elements not adjacent to the boundary as constraining only $[\boldsymbol{n}\cdot\nabla\times\psi^h]$ and $[\boldsymbol{n}\times\nabla\times\psi^h]$ specifies ψ^h only up to a constant.

We associate with (2.41) the least-squares principle: *find* $(\boldsymbol{\psi}^h,\omega^h,p^h)\in\widetilde{W}^h_r$ *such that*

$$\widetilde{J}^h_{-1}(\boldsymbol{\psi}^h,\omega^h,p^h;\boldsymbol{f})\le\widetilde{J}^h_{-1}(\boldsymbol{\phi}^h,\xi^h,q^h;\boldsymbol{f})\quad\forall(\boldsymbol{\phi}^h,\xi^h,q^h)\in\widetilde{W}^h_r\quad r>1 \tag{2.43}$$

where the approximating space is given by

$$\widetilde{W}^h_r=[V^{r+1}]\times V^r\times\breve{V}^r\,. \tag{2.44}$$

The least-squares principle for (2.42) is similar except that we allow for $r=1$ in the definition of the finite element space (2.44). Once (2.43) or its discrete negative norm companion are solved, the velocity is recovered using (2.40): on each element $\boldsymbol{u}^h|_K=\nabla\times\psi^h|_K$.

An alternative approach to the derivation of the dS-VP least-squares method is to start directly with a

stream-function vorticity formulation of the governing equations

$$\begin{cases} \nabla \times \omega + \nabla p = f & \text{on } \Omega \\ \nabla \times \nabla \times \psi - \omega = 0 & \text{on } \Omega \end{cases} \tag{2.45}$$

with boundary conditions

$$\begin{cases} \psi = 0 & \text{on } \partial\Omega \\ \boldsymbol{n} \times \nabla\psi = 0 & \text{on } \partial\Omega \end{cases} \tag{2.46}$$

The discontinuous formulation of the first order system defined in (2.45) and (2.46) results in the same functional as (2.41). For examples of various numerical methods based on this approach we refer to [9, 34, 37] and solution methods for the resulting equations are discussed in [32, 36]. Our approach is advantageous for two reasons. First, it clearly shows the connection with some of the most popular least-squares formulations for the Stokes equations. More importantly, our approach exposes the resulting dS-VP formulations as special cases of the discontinuous velocity LSFEMs with a specific choice of a *divergence-free* basis. In this section we define this basis through a stream function as in (2.40) primarily because of the simplicity of this choice; however, our approach can easily accommodates any choice of a divergence-free velocity basis. For examples of Discontinuous Galerkin methods that adhere to the latter strategy we refer to [29, 31], and the references therein.

Notably, the flux in the discrete least-squares method for the Darcy flow in two-dimensions [26] is approximated by a similar discontinuous space $\mathbf{V}^h = \nabla(V_D^h) \oplus \nabla \times (V_N^h)$ where V_D^h and V_N^h are standard C^0 finite element spaces constrained by zero on the Dirichlet and Neumann portions of the boundary. The key difference is that our approach deals with the discontinuity of the approximating space by including appropriate jump terms and retaining the original differential operators, whereas [26] retains the global inner products and switches to weak discrete differential operators defined using integration by parts.

2.5.1 Numerical studies

We now demonstrate the computational properties of the dS-VP formulation (2.41). We first estimate the numerical convergence rates of the method using manufactured solutions. We then estimate the mass conservation properties using the two test problems defined in Figures 2.1 and 2.2. We consider the finite element space(2.44) with $r = 2$, that is,

$$\widetilde{W}_2^h = [V^3] \times V^2 \times \breve{V}^2 \,. \tag{2.47}$$

h	$\|\psi - \psi^h\|_0$	rate	$\|\psi - \psi^h\|_1$	rate
1/2	4.555e-03		5.431e-02	
1/4	4.014e-04	3.50	6.886e-03	2.98
1/8	3.767e-05	3.46	1.004e-03	2.88
1/16	5.280e-06	3.27	1.340e-04	2.88
1/32	6.976e-07	3.16	1.711e-05	2.89

Table 2.1: *Error and convergence rate estimates for the stream function*

The approximation of the stream function by $[V^3]$ elements is consistent with the requirement[3] that the velocity in the parent least-squares formulation (2.23) should be approximated by at least $\mathbf{V}^2$ elements. Violation of this requirement has negative consequences for the accuracy of (2.23) [14]. To assess the relevance of this requirement for the dS-VP formulation, we include comparisons with an implementation of the new method which uses the equal order space

$$\widehat{W}_2^h = [V^2] \times V^2 \times \breve{V}^2 \,.$$

We assess the convergence rates of the method on a sequence of five uniform partitions of the unit square $[0,1] \times [0,1]$ into square elements with side lengths of $h_i = 2^{-i}$, $i = 1, \ldots, 5$. Suppose that e_i is the error corresponding to mesh-size h_i. We use incremental linear regression to generate a sequence of convergence rate estimates α_i, $i = 2, 3, 4, 5$. Specifically, α_i is the slope of the best least-squares fit to the data points $\{(-k, \log_2 e_k)\}_{k=1,i}$.

To generate the error data e_i we solve the dS-VP formulation (2.41) with a right hand side and boundary data corresponding to the exact solution

$$\psi = \cos(\pi x) + \cos(\pi y)\,, \tag{2.48}$$

$$\omega = \nabla \times \nabla \times \psi = \pi^2(\cos(\pi x) + \cos(\pi y))\,, \tag{2.49}$$

$$p = \cos(x)\exp(y)\,, \tag{2.50}$$

which leads to

$$\boldsymbol{f} = \begin{bmatrix} -\pi^3 \sin(\pi y) - \exp(y)\sin(x) \\ \pi^3 \sin(\pi x) + \exp(y)\cos(x) \end{bmatrix}. \tag{2.51}$$

Because the dS-VP formulation is derived from the weighted L^2 LSFEM (2.23), and the minimal approximation condition is satisfied, we anticipate convergence rates for the vorticity and the pressure to be at least as predicted by Theorem 2.3.1 for $r = 2$. However, our implementation uses vorticity and pressure

[3] We note that this minimal approximation condition does not extend to the negative norm LSFEM (2.30).

h	$\|\omega - \omega^h\|_0$	rate	$\|\omega - \omega^h\|_1$	rate
1/2	1.216e+00		1.072e+01	
1/4	1.079e-01	3.49	1.600e+00	2.74
1/8	1.200e-02	3.33	3.257e-01	2.52
1/16	1.486e-03	3.22	8.037e-02	2.35
1/32	1.938e-04	3.14	2.258e-02	2.21

Table 2.2: *Error and convergence rate estimates for the vorticity*

h	$\|p - p^h\|_0$	rate	$\|p - p^h\|_1$	rate
1/2	1.895e+00		1.054e+01	
1/4	1.750e-01	3.44	1.446e+00	2.87
1/8	1.676e-02	3.41	2.688e-01	2.65
1/16	1.913e-03	3.32	6.188e-02	2.47
1/32	2.529e-04	3.23	1.515e-02	2.34

Table 2.3: *Error and convergence rate estimates for the pressure*

spaces of one degree higher than in the statement of the theorem. As a result, it is reasonable to expect that

$$\|p - p^h\|_0 = \|\omega - \omega^h\|_0 = O(h^3) \quad \text{and} \quad \|p - p^h\|_1 = \|\omega - \omega^h\|_1 = O(h^2)\,.$$

The rates of convergence for the stream function cannot be inferred directly from the theorem. Nonetheless, knowing that the dS-VP formulation originates in the optimally accurate and well-posed LSFEM (2.23), we anticipate that convergence rates for this variable will be close to the best approximation theoretic rates for V^3 elements.

These conjectures are largely confirmed by the data in Tables 2.1 – 2.3, except for the L^2 rate of the stream function which is less than the expected value of 4. However, as the mesh is refined, the H^1-seminorm error rate for this variable approaches the best theoretical value of 3. As a rule, L^2 rates tend to be less reliable and so further theoretical studies are necessary to establish the convergence of the new dS-VP formulation. Nevertheless, the preliminary convergence results reported here are encouraging and suggest that the dS-VP formulation is optimally accurate.

We use the backward-facing step and the flow past a cylinder test problems shown in Figure 2.1 and Figure 2.2, respectively. First, we compare and contrast the mass loss in (2.23) and (2.41) using the same grids as in Section 2.3.1: 900 rectangular elements for the backward-facing step; 1296 triangular elements for the flow past a cylinder with $r = 0.6$ and 1104 triangular elements for $r = 0.9$. We also investigate the relevance of the minimal approximation condition for the mass conservation in the dS-VP formulation by comparing the mass losses in implementations of (2.41) with $[V^3]$ and $[V^2]$ elements for the stream function, respectively. Our final study examines improvement in the mass conservation under mesh refinement.

Recall that in the dS-VP formulation the normal component of the velocity boundary condition is prescribed through an equivalent Dirichlet condition on the stream function, and that the tangential component is enforced weakly by including its residual in (2.41). In the case of the backward-facing step the velocity boundary condition is given by (2.31). On Γ_{in} and Γ_{out} velocity is only a function of y and $\boldsymbol{u} \cdot \boldsymbol{n} = \pm u_1$. Integration of u_1 along Γ_{in} and Γ_{out} yields an equivalent Dirichlet boundary condition on the stream function:

$$\boldsymbol{\psi}_{in} = -\frac{8}{3}y^3 + 6y^2 - 4y + C_1 \quad \text{and} \quad \boldsymbol{\psi}_{out} = \frac{y^2}{2} - \frac{y^3}{3} + C_2\,.$$

The constants C_1 and C_2 are chosen so that $\boldsymbol{u}_{in}(0.5) = \boldsymbol{u}_{out}(0)$ and $\boldsymbol{u}_{in}(1) = \boldsymbol{u}_{out}(1)$. On the top and the bottom walls ψ is set to a constant value equal to $\boldsymbol{u}_{in}(1)$ and $\boldsymbol{u}_{in}(0.5)$, respectively.

For the flow past a cylinder the velocity boundary condition is specified in (2.32). An equivalent Dirichlet condition on the stream function, which prescribes the same normal velocity component is given by

$$\psi_{in} = \psi_{out} = \psi_{wall} = y - \frac{y^3}{3}\,.$$

Because $\boldsymbol{u} \times \boldsymbol{n} = 0$ on $\partial\Omega$ for both test problems, accounting for this part of the velocity boundary condition does not require additional terms beyond adding its residual, written as $(\nabla \times \psi) \times \boldsymbol{n}$, to the least-squares functional (2.41).

Results from our first study are summarized in Figure 2.5. The mass losses in the new dS-VP formulation (2.41) are compared to its parent LSFEM (2.23) for the backward-facing step and for the flow past a cylinder with $r = 0.6$ and $r = 0.9$. In all three cases the dS-VP solution shows significant improvements in the mass conservation, as measured by the percent mass loss formula (2.33). For the backward-facing step, the maximum mass loss is less than 0.5% with most of the mass loss localized at the reentrant corner. On the rest of the domain, the solution is basically conserved over any closed subdomain. For the flow past a cylinder, the center and right panels in Figure 2.5 reveal that the mass loss in the new dS-VP formulation does not deteriorate as the radius of the cylinder increases from 0.6 to 0.9. In the narrowest region of the computational domain the global mass in the dS-VP solution fluctuates within less than 1%, and in the rest of the domain it is essentially constant. These results clearly show that mass conservation in the new dS-VP formulation is superior to that of the weighted L^2 (2.23) *and* the discontinuous velocity LSFEM (2.37).

Figure 2.6 shows the results of our second study, which compares conservation properties of (2.41) implemented with $[V^3]$ and $[V^2]$ elements for the stream function, respectively. The objective is to determine whether the dS-VP formulation inherits the minimal approximation condition from its parent LSFEM (2.23) as a strong, dominant trait. If this were the case, then an dS-VP implementation employing the equal order

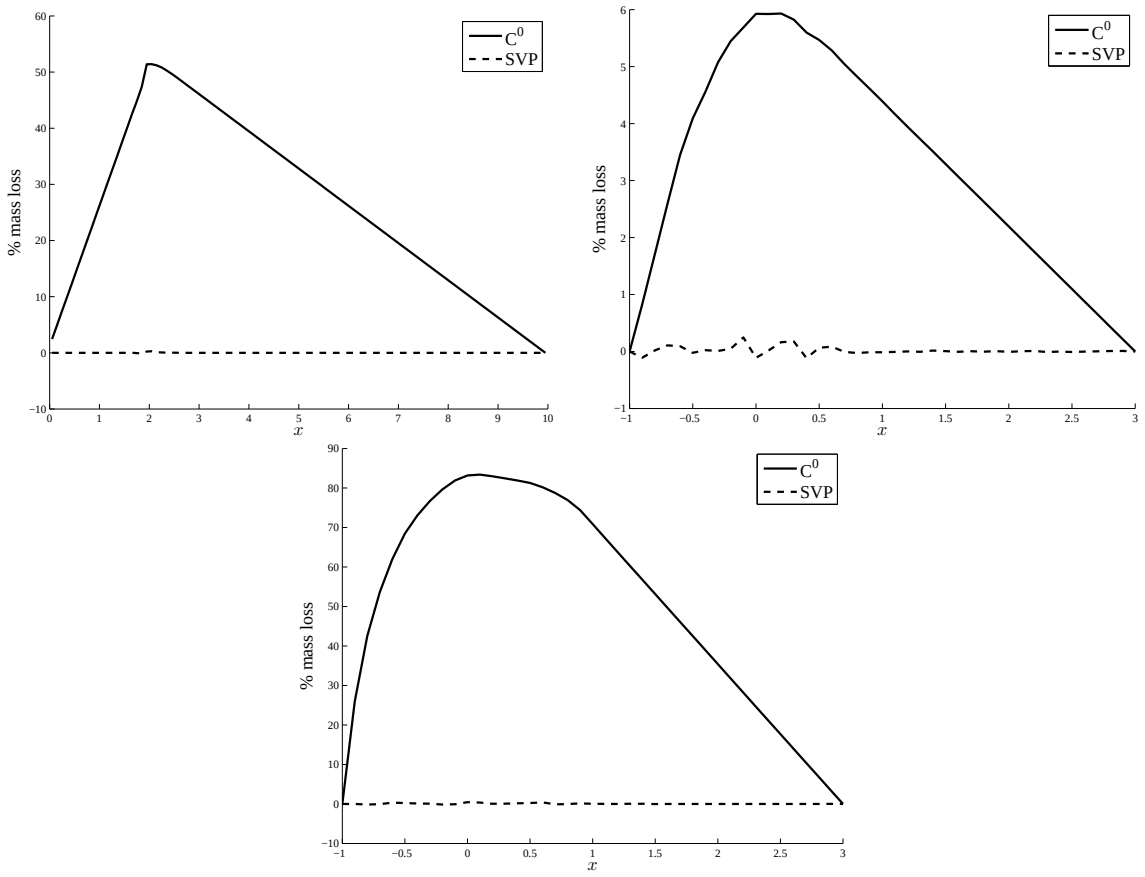


Figure 2.5: *Comparison of mass loss in (2.23) and (2.41) for the backward-facing step (left panel) and the flow past a cylinder with $r = 0.6$ (center panel) and $r = 0.9$ (right panel). Solid line represents the weighted L^2 formulation (2.23), dashed line is the new dS-VP formulation (2.41). Values are computed using (2.33) along vertical lines placed at every 0.1 units along the x-axis. A total of 100 lines are used for the backward-facing step and 40 lines are used for the flow past a cylinder.*

space $\widehat{W}_2^h = [V^2] \times V^2 \times \check{V}^2$ should experience noticeable deterioration in the mass conservation. However, the plots in Figure 2.6 suggest that this is not the case, and that the dS-VP formulation continues to exhibit high performance with an $[V^2]$ stream function. For the backward-facing step the switch from $[V^3]$ to $[V^2]$ elements causes the maximum mass loss to grow from 0.5% to 1.09%. For the flow past a cylinder with $r = 0.6$ the maximum loss grows from 0.3% to 0.8%, and for $r = 0.9$ we see the greatest growth from 0.4% to 2%. However, even with these increases, the mass loss in all three test problems remains within acceptable limits and well below the mass losses in (2.23) and (2.37). Based on these results we conclude that the minimal approximation condition is not a principal limitation for the dS-VP formulation as it is for its C^0 parent (2.23). The possibility to implement (2.41) with equal order elements without serious deterioration in accuracy is valuable from an efficiency standpoint because such elements have more uniform data structures.

Our third study examines improvement in mass conservation under refinement. For this study the

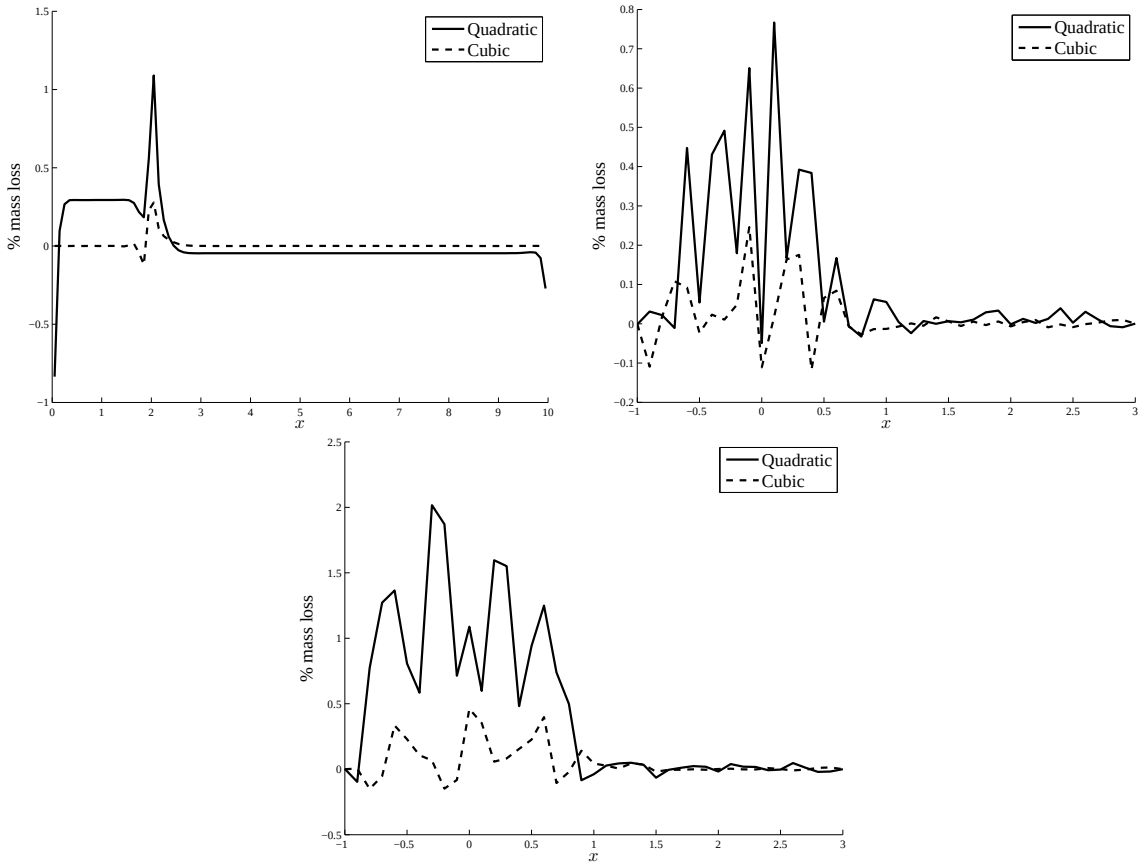


Figure 2.6: *Comparison of mass loss in (2.41) implemented with $[V^2]$ and $[V^3]$ elements for the stream function, for the backward-facing step (left panel) and the flow past a cylinder with $r = 0.6$ (center panel) and $r = 0.9$ (right panel). Solid line represents implementation of (2.41) with an $[V^2]$ stream function, dashed line corresponds to an $[V^3]$ stream function. Values are computed using (2.33) along vertical lines placed at every 0.1 units along the x-axis. A total of 100 lines are used for the backward-facing step and 40 lines are used for the flow past a cylinder.*

original finite element partitions for the backward-facing step and for the flow past a cylinder problems were uniformly refined to grids with four times as many elements. Thus, the refined grids comprise of 3600 rectangular elements for the backward-facing step, 5184 triangle elements for the flow past a cylinder of radius 0.6, and 4416 triangle elements for the flow past a cylinder with radius 0.9. Results of the refinement study are shown in Figure 2.7. The most significant improvement occurs in the backward-facing step problem where the maximum mass loss decreases almost five-fold from 0.28% to 0.06%. We see the same improvement in the flow past a cylinder of radius 0.6. The reduction in the maximum mass loss when $r = 0.9$ is somewhat smaller, but still valuable. The important conclusion from this study is that mesh refinement consistently delivers further improvements to the mass conservation of the dS-VP formulation.

Additionally, there are some qualitative differences in the finite element solutions computed by (2.41)

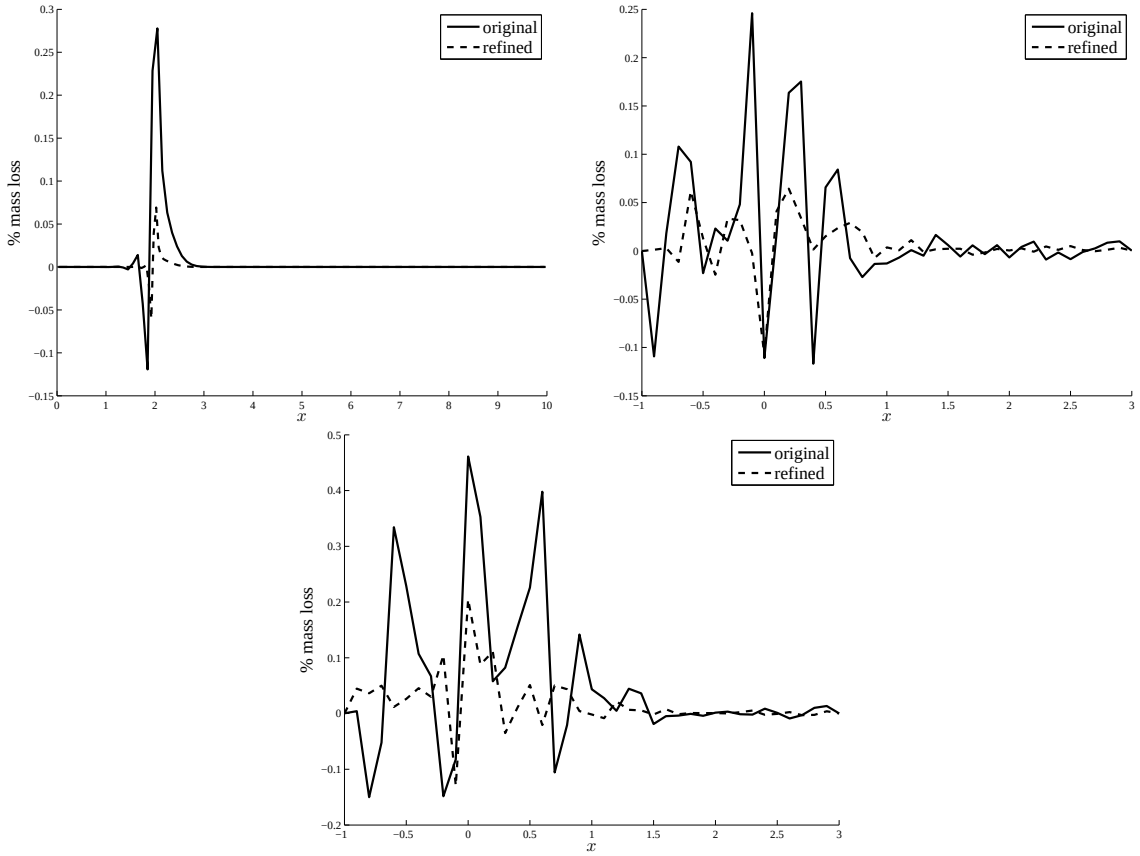


Figure 2.7: *Improvement of the mass conservation in (2.41) under mesh refinement for the backward-facing step (left panel) and the flow past a cylinder with $r = 0.6$ (center panel) and $r = 0.9$ (right panel). The solid and the dashed line represent the dS-VP formulation (2.41) on the original and on the refined meshes, respectively. Values are computed using (2.33) along vertical lines placed at every 0.1 units along the x-axis. A total of 100 lines are used for the backward-facing step and 40 lines are used for the flow past a cylinder.*

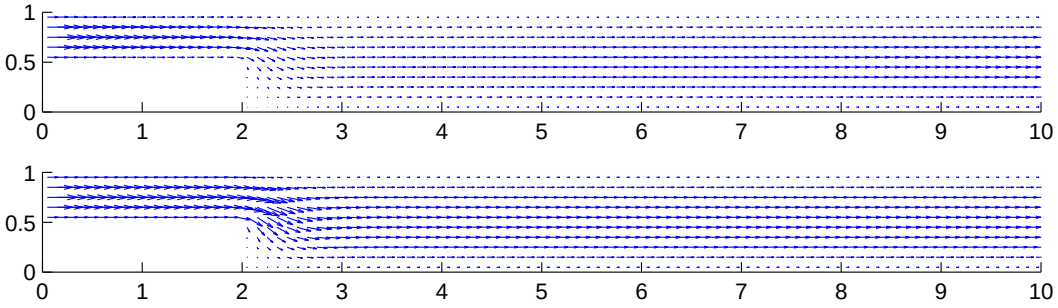


Figure 2.8: *Velocity plot of the weighted L^2 LSFEM (2.23) (top) and the dS-VP formulation (2.41) (bottom) for the backward-facing step.*

and (2.23), which can be attributed to the mass losses in the latter. Plots of the velocity, pressure, and vorticity for the backward-facing step problem computed by these two methods are compared in Figures 2.8–

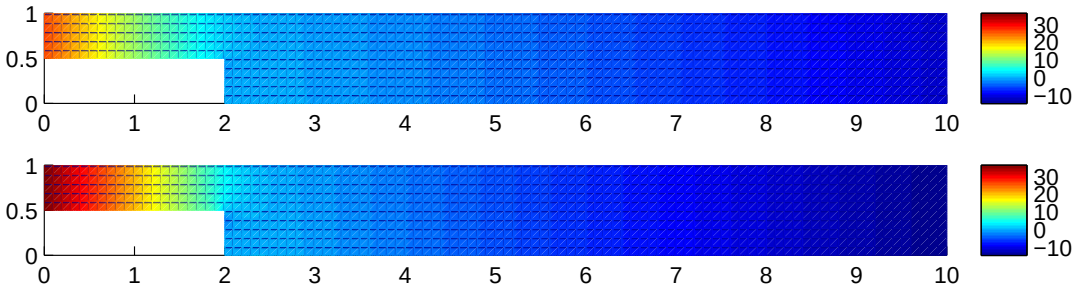


Figure 2.9: *Pressure plot of the weighted L^2 LSFEM (2.23) (top) and the dS-VP formulation (2.41) (bottom) for the backward-facing step.*

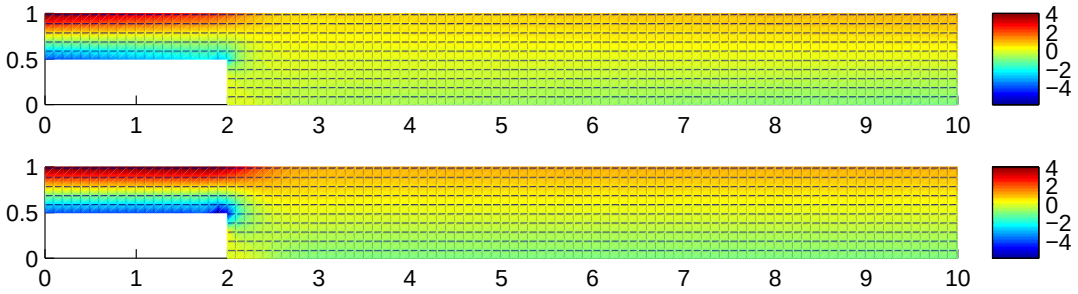


Figure 2.10: *Vorticity plot the weighted L^2 LSFEM (2.23) (top) and the dS-VP formulation (2.41) (bottom) for the backward-facing step.*

2.10. One significant difference between the two solutions is seen in the velocity plots shown in Figure 2.8. The SVP solution exhibits the expected behavior near the reentrant step and maintains the characteristic parabolic velocity profile throughout the full length of the problem domain. In contrast, the severe mass loss in the solution of (2.23) near the reentrant step leads to an underestimate of the velocity magnitude and weakening of its parabolic profile in this region.

Plots of the finite element solutions for (2.41) and (2.23) in the case the flow past a cylinder of radius 0.6 are compared in Figures 2.11–2.13. The computed velocity fields by (2.41) and (2.23) are similar for this example since the maximum mass loss in the case of (2.23) is only 6%. However, even for this case of low mass loss, the inadequate pressure drop in the region behind the cylinder is noticeable, as depicted in Figure 2.12. Visible differences are also evident in the vorticity plots shown in Figure 2.13.

However, setting the cylinder radius to 0.9 intensifies the difficulty for (2.23), which now exhibits a loss of over 80% of the mass in the narrowest region—see Figure 2.3. Accordingly, the qualitative differences between the solutions of (2.41) and (2.23) become more pronounced, especially for the velocity and the pressure. Because the cylinder restricts 90% of the channel, the fluid velocity must increase significantly in the regions between the boundary walls and the top and the bottom of the cylinder. As shown in Figure 2.14,

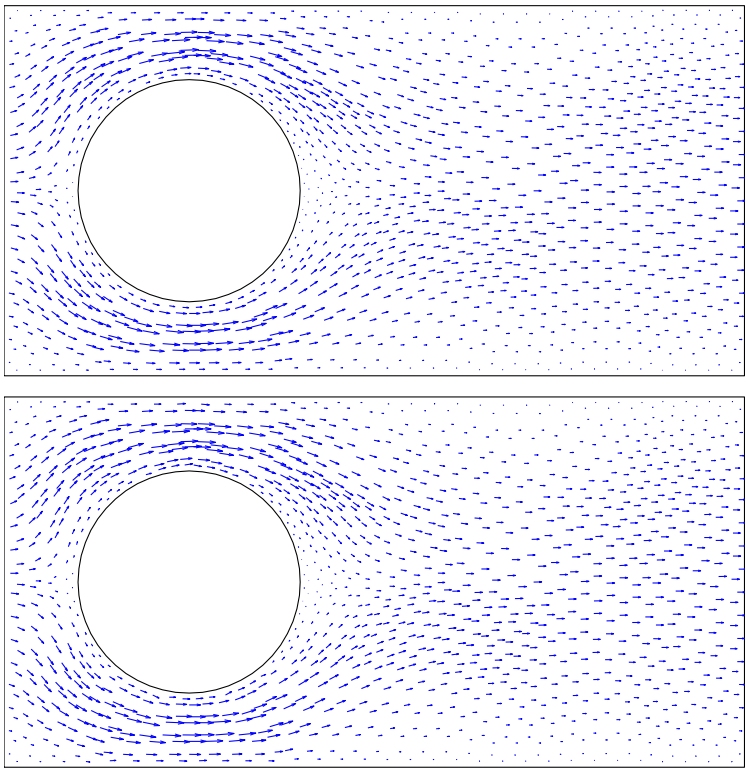


Figure 2.11: *Velocity plot of the weighted L^2 LSFEM (2.23) (top) and the dS-VP formulation (2.41) (bottom) for the flow past a cylinder with $r = 0.6$.*

the SVP solution demonstrates this behavior. In contrast, the magnitude of the velocity in the solution of (2.23) is comparable to that of the inflow boundary, thus underestimating the velocity. In Figure 2.15 we also observe that the pressure drop behind the cylinder in the solution to (2.23) is underestimated. Moreover, as before, the visible qualitative differences extend to the vorticity plots in Figure 2.16.

In this section we have introduced a locally divergence free formulation for the Stokes equations through the use a local stream function. We showed that the mass conservation of the formulation is much improved over the standard C^0 method. A few disadvantages with the dS-VP formulation are the implicit definition of the velocity boundary conditions through the boundary conditions on the stream function. Given the velocity boundary conditions, it was necessary to integrate the boundary condition to find the equivalent stream function boundary condition. For curved domains, the translation from velocity boundary conditions to stream function boundary conditions is non trivial. Also, due to the fact that the stream function is a scalar function while the velocity is a vector field, Dirichlet boundary conditions on the stream function sets only one component of the velocity field, the other component is set weakly. Due to the two derivatives on the stream function in the definition of the vorticity, the matrix exhibits high condition numbers that scale

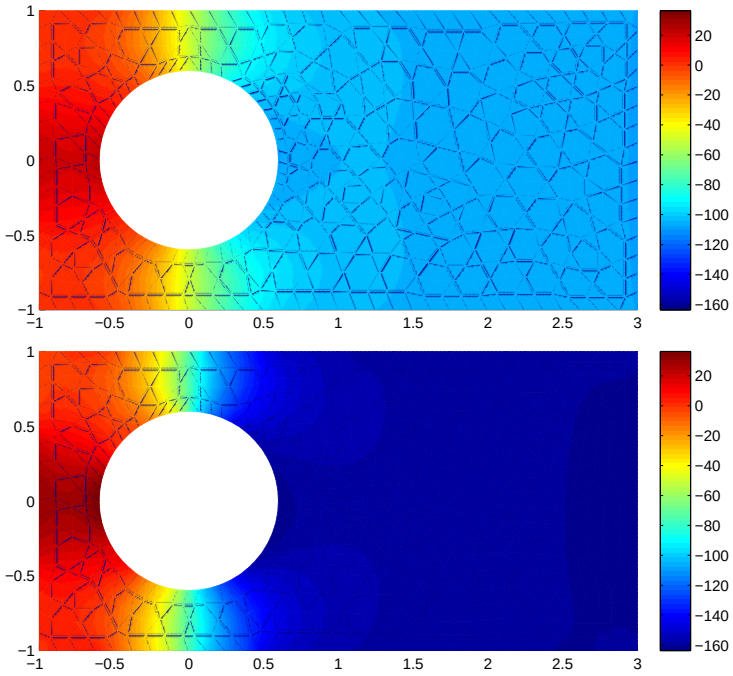


Figure 2.12: *Pressure plot of the weighted L^2 LSFEM (2.23) (top) and the dS-VP formulation (2.41) (bottom) for the flow past a cylinder $r = 0.6$.*

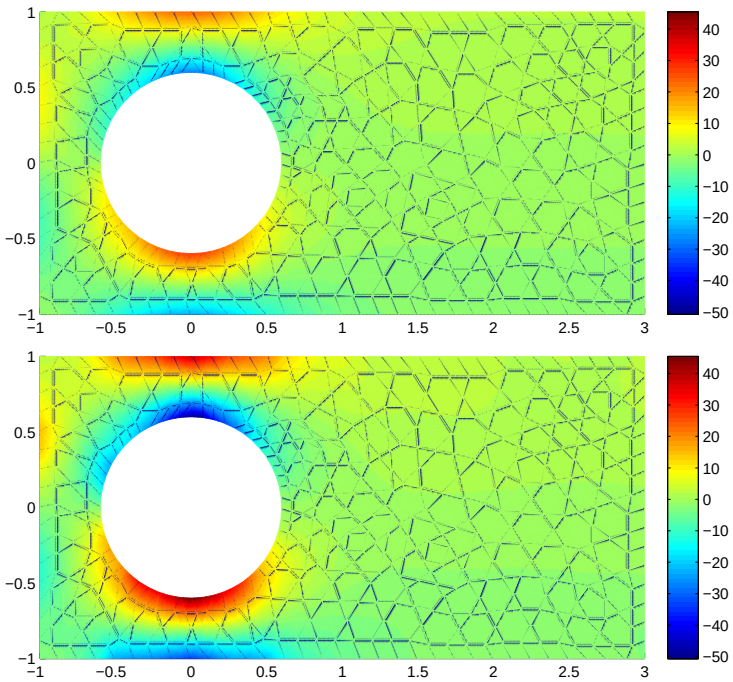


Figure 2.13: *Vorticity plot of the weighted L^2 LSFEM (2.23) (top) and the dS-VP formulation (2.41) (bottom) for the flow past a cylinder $r = 0.6$.*

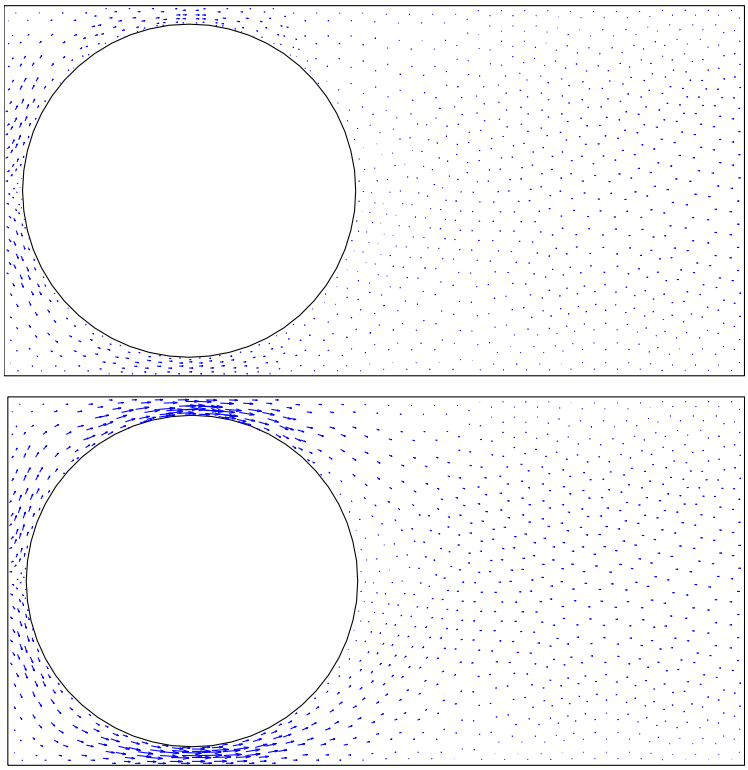


Figure 2.14: *Velocity plot of the weighted L^2 LSFEM (2.23) (top) and the dS-VP formulation (2.41) (bottom) for the flow past a cylinder with $r = 0.9$.*

poorly with respect to mesh size. Thus, it would be useful to eliminate the use of the stream function and instead, use a divergence free velocity formulation. We discuss such methods in Section 2.6.

2.6 Divergence free velocity-vorticity-pressure

This section is aimed at improving the dS-VP introduced in Section 2.5. This work can be found in [23]. We formulate a new locally conservative LSFEM for the velocity-vorticity-pressure first order system by using a piecewise divergence free velocity basis from [5]. The second order terms of the least-squares functional (2.41) are eliminated and imposition of the velocity boundary condition is simplified. Furthermore, the minimal admissible polynomial degree is reduced from cubic for the stream function in the dS-VP formulation to quadratic. Due to the lower order basis, the dimension of the velocity approximating space is approximately reduced by half reducing the size of the matrix.

We introduce additional test problems in Section 2.6.1. and in Section 2.6.2, we present the new dV-VP

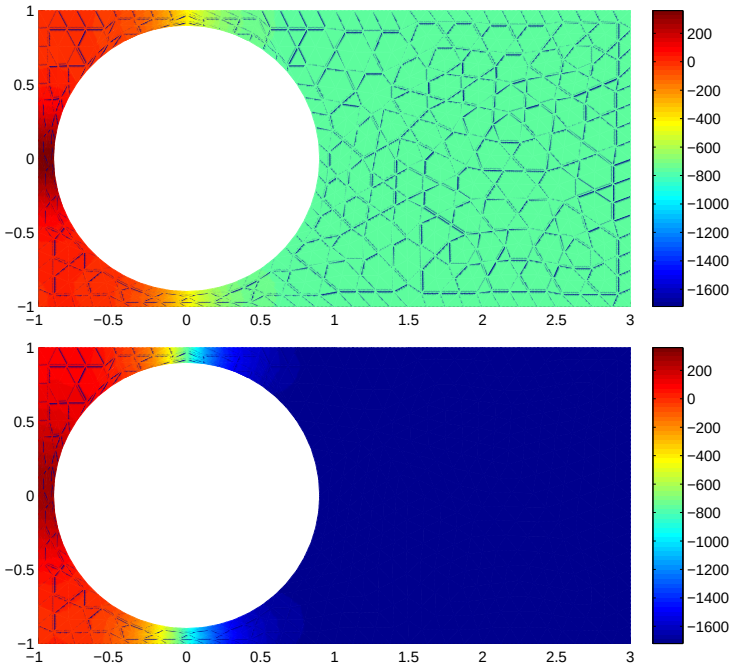


Figure 2.15: *Pressure plot of the weighted L^2 LSFEM (2.23) (top) and the dS-VP formulation (2.41) (bottom) for the flow past a cylinder $r = 0.9$.*

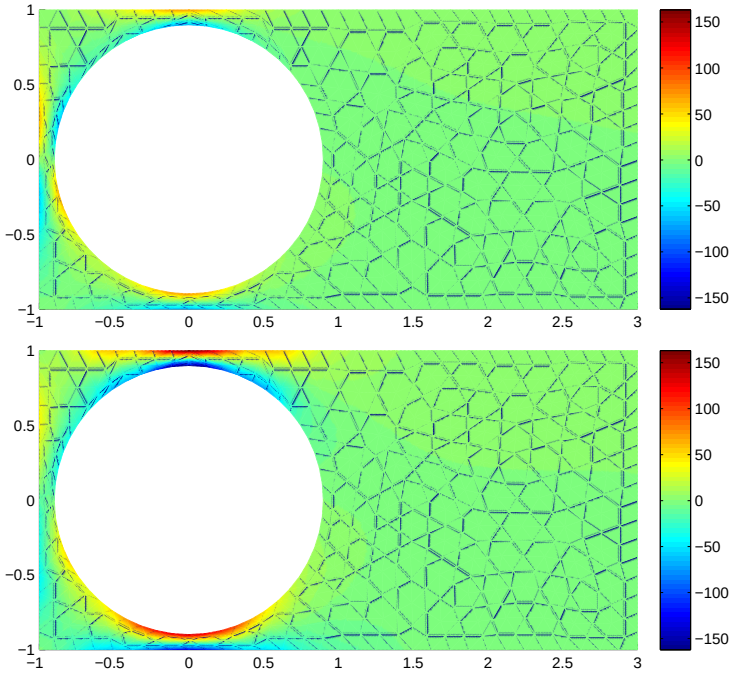


Figure 2.16: *Vorticity plot of the weighted L^2 LSFEM (2.23) (top) and the dS-VP formulation (2.41) (bottom) for the flow past a cylinder $r = 0.9$.*

LSFEM by introducing a series of intermediate functionals, as outlined in the following:

$$J^S_{(h)} \xrightarrow{\psi \text{ to } \boldsymbol{u}} J^V_{(h)} \xrightarrow{\text{modify jump}} \widehat{J}^V_{(h)} \xrightarrow{\text{implicit } \psi} \widetilde{J}^V_{(h)}, \tag{2.52}$$

where $J^S_{(h)}$ is the dS-VP functional from Section 2.5. We then introduce $J^V_{(h)}$, wherein the stream function is replaced by a divergence-free velocity basis, followed by $\widehat{J}^V_{(h)}$ in which jump terms are used to accommodate for the difference in scaling of the divergence-free velocity basis, and finally we use $\widetilde{J}^V_{(h)}$ to enforce global continuity of an implicit stream function. In Section 2.6.4 we define a diagonal preconditioner for the discrete problems and in Section 2.6.5 we focus on computational studies of the dV-VP formulation, which includes convergence rates, conservation of mass, preconditioning, and impact of the divergence-free basis choice on the properties of the LSFEMs.

2.6.1 Test problems

In addition to the test problems used in Section 2.5, we introduce additional test problems. In order to keep the mass loss computations comparable between test domains, each mesh is well refined and generated by using an average element size of $h \approx 0.03 - 0.04$. We reuse the backward facing step described in Figure 2.1 with boundary conditions (2.31). The domain is discretized using 6442 triangles. The second test problem from Section 2.5, channel flow with cylindrical cutout, is also used with boundary conditions (2.32). We consider only the more difficult case with $r = 0.9$ and partitioned into 6011 triangles.

Two additional test problems that we introduce are the split channel and the restricted channel. In the split channel test, we model channel flow split into two separate channels and then finally combining back into a single channel. The computational domain begins with a height of 1 and splits off into two channels of height 0.5. For the boundary conditions, we set

$$\boldsymbol{u}_{in} = \boldsymbol{u}_{out} = \begin{bmatrix} (0.5 - y)(0.5 + y) \\ 0 \end{bmatrix}, \text{ and } \boldsymbol{u}_{wall} = \boldsymbol{0}. \tag{2.53}$$

This test problem is solved on $\mathcal{K}_h$ with 6694 triangles.

For the restricted channel domain, we have a channel flow that is pinched in on the top and bottom sides. The domain is the rectangular domain $[-2, 2] \times [-1, 1]$. The channel is pinched in at $x = 0$ using two semi-cylindrical cut outs of radius r. Similar to the cylinder flow domain, the larger the radius, the more narrow the opening of the channel and hence increasing the difficulty of the problem. In our examples, we use $= 0.9$. The boundary conditions are set as in (2.32) and the domain is meshed using 4124 triangles.

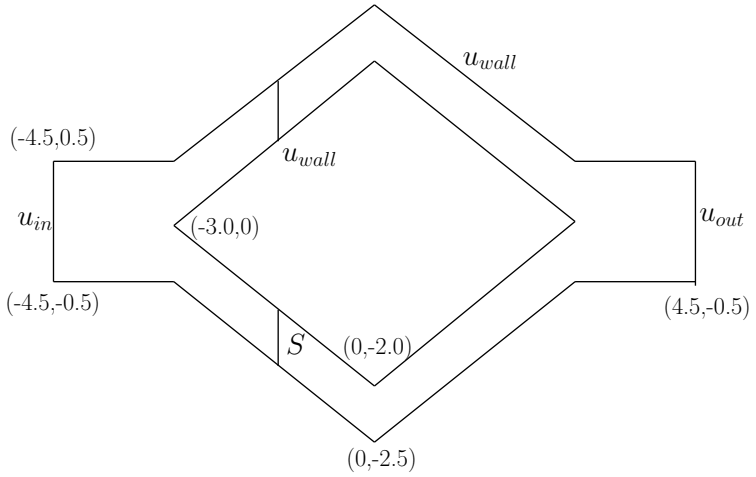


Figure 2.17: *Geometry of split channel test problem.*

In each of the test problems the boundary conditions are compatible with $\nabla \cdot \boldsymbol{u} = 0$. To assess mass conservation we follow the procedure from [22] and measure the total mass flow across a sequence of vertical surfaces connecting the top and the bottom sides of the computational domain. The lines denoted by S in Figures 2.1-2.18 show examples of such surfaces.

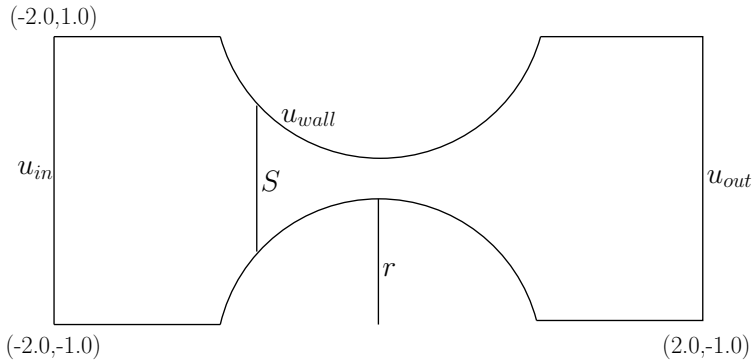


Figure 2.18: *Geometry of restricted channel test problem.*

2.6.2 Formulation

Piecewise divergence free velocity element

In this section we introduce a piecewise solenoidal velocity element $\mathbf{D}_r$, with $r \geq 1$, as proposed in [5]. The dimension of $\mathbf{D}_r$ depends only on the polynomial degree r and not on the shape of the reference element $\widehat{\kappa}$.

For example the linear piecewise solenoidal space in two dimensions is

$$\mathbf{D}_1(\widehat{\kappa}) = \left\{ \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} y \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ x \end{pmatrix}, \begin{pmatrix} x \\ -y \end{pmatrix} \right\}, \tag{2.54}$$

while the quadratic space is

$$\mathbf{D}_2(\widehat{\kappa}) = \mathbf{D}_1(\widehat{\kappa}) \cup \left\{ \begin{pmatrix} y^2 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ x^2 \end{pmatrix}, \begin{pmatrix} x^2 \\ -2xy \end{pmatrix}, \begin{pmatrix} -2xy \\ y^2 \end{pmatrix} \right\}. \tag{2.55}$$

In d-dimensions, we arrive at

$$\dim \mathbf{D}_r(\widehat{\kappa}) = \frac{d(d+r)! - (d+r-1)!r}{d!r!}\,.$$

We define the full velocity space $\mathbf{D}_r(\Omega)$ by translation and scaling of the reference element space

$$\mathbf{D}_r(\Omega) = \{\boldsymbol{v}_h \in \mathbf{L}^2(\Omega) \,|\, \boldsymbol{v}_h(\boldsymbol{x})|_\kappa = \widehat{\boldsymbol{v}}_h(\boldsymbol{x} - \boldsymbol{b}_\kappa)/J_\kappa^{(\deg \widehat{\boldsymbol{v}})/2};\ \widehat{\boldsymbol{v}}_h \in \mathbf{D}_r(\widehat{\kappa})\}, \tag{2.56}$$

where $\deg \widehat{\boldsymbol{v}}$ is the polynomial degree of basis function $\widehat{\boldsymbol{v}}_h$ and $\boldsymbol{b}_\kappa$ is the barycenter of element κ. We choose such a scaling so each basis function is normalized to unity on κ.

Standard finite elements on quasi-uniform grids satisfy the following inequalities.

Approximation. For every $v \in H^{r+1}(\Omega)$ there exists $I(v) \in V^r(\Omega)$ such that

$$\|v - I(v)\|_0 + h\|v - I(v)\|_1 \leq Ch^{r+1}\|v\|_{r+1}, \tag{2.57}$$

where C is independent of h.

Inverse inequalities. There exists positive constants C_1 and C_2, independent of h, such that for every element $\kappa \in \mathcal{K}_h$

$$C_1 h^2 |\vec{v}|^2 \leq \|v_h\|_{0,k} \leq C_2 h^2 |\vec{v}|^2\,. \tag{2.58}$$

Additionally, finite element functions satisfy the inverse inequalities

$$\|v_h\|_{1,\kappa} \leq Ch^{-1}\|v_h\|_{0,\kappa} \quad \text{and} \quad \|v_h\|_{1/2,e} \leq Ch^{-1/2}\|v_h\|_{0,e} \tag{2.59}$$

These inequalities hold whenever the mesh is quasi-uniform and the finite element spaces are defined by transformation of a reference space [35, Lemma 9.7, p.386; Lemma 1.138, p.75]. These inequalities were

used in the dS-VP formulation and are required to maintain the proper scaling of these mesh dependent terms.

However, the varying polynomial degrees of the basis functions in $\mathbf{D}_r(\widehat{\kappa})$ prevent (2.58) and (2.59) from holding. By using translation and mesh-dependent scaling proportional to the polynomial degree of each basis function we are able to define piecewise solenoidal bases for $\mathbf{D}_r(\Omega)$ that satisfy inverse inequalities. We note that this is similar to the piecewise divergence free basis defined in [30], but uses a different scaling for which the mass matrix is not spectrally equivalent to a scaled identity.

The velocity space (2.56) is completely discontinuous and is not H^1-conforming, yet $\mathbf{D}_r(\Omega)$ exhibits an optimal approximation property [5, Theorem 4.3]: for every $\boldsymbol{v} \in \mathbf{H}^{r+1}(\kappa)$ there exists $I(\boldsymbol{v}) \in \mathbf{D}_r(\kappa)$ such that

$$\|\boldsymbol{v} - I\boldsymbol{v}\|_{j,\kappa} \leq Ch^{r+1-j}|\boldsymbol{v}|_{r+1,\kappa};\ j = 0, \ldots, r\,. \tag{2.60}$$

For examples of Discontinuous Galerkin methods, which use $\mathbf{D}_r$ elements we refer to [30, 44] and the references therein. The paper [5] also compares $\mathbf{D}_r$ elements with other nonconforming spaces such as Crouzeix-Raviart elements [49].

Our formulation of the Stokes equations using the divergence free basis follows the same approach as in Section 2.5. We begin with the continuous functional (2.30) and replace the velocity space with the divergence free basis $\mathbf{D}_r$ defined in (2.60). Therefore, the approximating space changes from (2.25) to

$$X_r^h = \mathbf{D}^r \cap \mathbf{H}_0^1(\Omega) \times V^{r-1} \times V^{r-1} \cap L_0^2(\Omega), \quad r > 1 \tag{2.61}$$

and its equal order counterpart

$$X_r^h = \mathbf{D}^r \cap \mathbf{H}_0^1(\Omega) \times V^r \times V^r \cap L_0^2(\Omega), \quad r > 1. \tag{2.62}$$

A straightforward dimensional analysis shows that for the solenoidal vector fields in (2.56) and standard nodal functions $\psi_h \in [V^r](\Omega)$ we have

$$\int_\varepsilon [\boldsymbol{u}_h]^2 dl = O(h) \quad \text{and} \quad \int_\varepsilon [\nabla \times \psi_h]^2 dl = O(h^{-1}), \tag{2.63}$$

for some edge $\varepsilon \in \mathcal{E}_h$. Therefore, in order to preserve the relative scaling of the terms in the dS-VP functional (2.41) when using the piecewise solenoidal space (2.56) it is necessary to change the weight of the velocity jump term from h^{-1} to h^{-3}. Taking this and the divergence-free property of the velocity basis into

consideration, we introduce a new functional

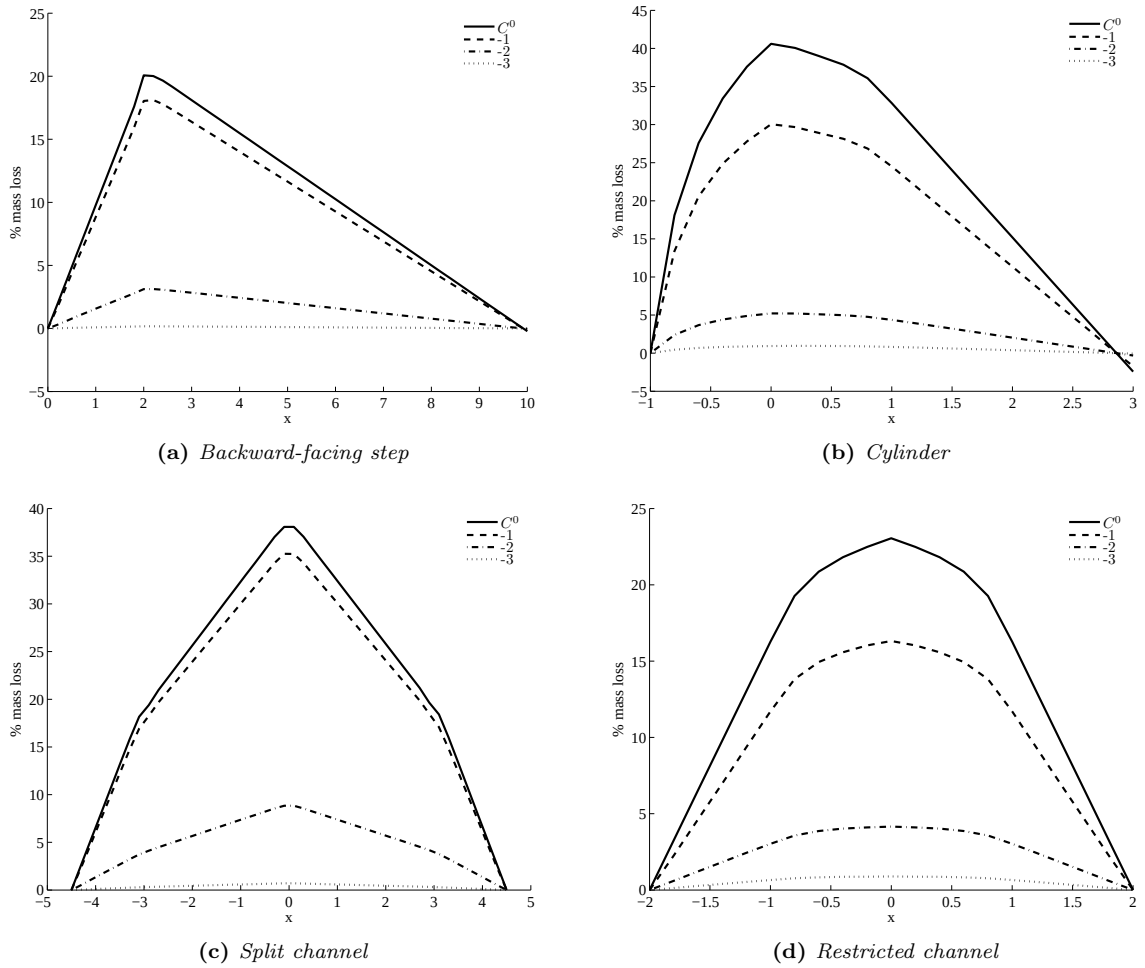
$$\widehat{J}^{V,\alpha}_{(h)}(\boldsymbol{u}_h, \omega_h, p_h; \boldsymbol{f}_h) = \\ \|\nabla \times \omega_h + \nabla p_h - \boldsymbol{f}_h\|^2_{(h)} + \sum_{\kappa \in \mathcal{K}_h(\Omega)} \|\nabla \times \boldsymbol{u}_h - \omega_h\|^2_{0,\kappa} + \sum_{\varepsilon \in \mathcal{E}_{h,0}} h^{-\alpha} \|[\![\boldsymbol{u}_h]\!]\|^2_{0,\varepsilon} \tag{2.64}$$

and (2.37) becomes

$$\widehat{J}^{V}_{(h)}(\boldsymbol{u}_h, \omega_h, p_h; \boldsymbol{f}_h) := \widehat{J}^{V,-3}_{(h)}(\boldsymbol{u}_h, \omega_h, p_h; \boldsymbol{f}_h) \tag{2.65}$$

To demonstrate the role of proper weighting of the velocity jump we solve the two test problems using three different weights for this term in (2.64). Our implementation uses the equal order space (2.62) with $r = 2$. We set $\|\cdot\|_{(h)} = h\|\cdot\|_0$ and choose $\alpha = -1, -2, -3$. The C^0 least-squares solution of (2.24), implemented with the equal order space $X^{(2)}_h$, provides the benchmark. Figure 2.19 demonstrates that proper weighting of this term significantly reduces the mass loss in the least-squares solution. Yet, it also shows that if the changes in the scaling of the least-squares terms induced by the piecewise solenoidal velocity space (2.56) are not taken into consideration, conservation of mass suffers. Specifically, if the weight of the velocity jump is left at h^{-1}, as in the dS-VP functional (2.41), then the peak mass loss in all four test problems is similar to the C^0 solution.

Figure 2.19 demonstrates that when using the correct weight on the jump term, (2.64) performs very well — i.e., the maximum mass loss in each test problem is less than 1% at 0.17%, 0.95%, 0.70%, and 0.88% for each test problem, respectively. However, we remark that the meshes used in the test cases are very well refined. On less refined meshes the mass loss for (2.64) is more evident. For example, the plots of mass loss in Figure 2.20 demonstrate that on a less refined mesh ($h \approx 0.07$), the maximum mass loss is around 2% even with the jump weight set at h^{-3} despite the piecewise solenoidal basis for the velocity field. When compared with the dS-VP formulation (2.41), the method exhibits considerably more mass loss. Because the exact velocity is divergence free, there is a scalar stream function ψ such that $\boldsymbol{u} = \nabla \times \psi$. The piecewise solenoidal velocity space $\mathbf{D}_r(\Omega)$ has this property locally — i.e., if $\boldsymbol{v}_h \in \mathbf{D}_r(\Omega)$. Thus, on every element $\kappa \in \mathcal{K}_h$, there is an implicit stream function ψ_κ such that $\boldsymbol{v}_h|_\kappa = \nabla \times \psi_\kappa$. Yet, the existence of an implicit stream function ψ_κ on each element does not imply that a piecewise solenoidal field $\boldsymbol{v}_h \in \mathbf{D}_r(\Omega)$ approximates the curl of a *global* stream function. This requires the implicit stream functions ψ_κ on adjacent elements to be nearly equal along the interfaces between the elements, and motivates the construction of a such a function.



(a) *Backward-facing step* **(b)** *Cylinder*

(c) *Split channel* **(d)** *Restricted channel*

Figure 2.19: *Comparison of the mass loss in the discontinuous velocity LSFEM (2.64) with* $\|\cdot\|_{(h)} = h\|\cdot\|_0$*, and* $\alpha = -1, -2, -3$ *vs. standard* C^0 *LSFEM (2.23).*

2.6.3 Implicit stream function

Since we related $\boldsymbol{u} = \nabla \times \psi$, the jump in velocity in (2.65) only controls the continuity of $\nabla \times \psi_k$, and does not directly "glue" ψ_k across element interfaces. To enforce this on the implicit stream functions, we propose to augment (2.65) with terms that imitate the jumps of the discontinuous stream function in (2.41).

For simplicity, we express the main idea using the trapezoidal rule to approximate the line integrals in these jumps. Let $V_0 = V_0(\varepsilon)$ and $V_1 = V_1(\varepsilon)$ be the endpoints of edge $\varepsilon \in \mathcal{E}_h$. Then,

$$\int_{\varepsilon} [\psi_h]^2 d\ell \approx \frac{|\varepsilon|}{2}\Big([\psi_h(V_0)]^2 + [\psi_h(V_1)]^2\Big). \tag{2.66}$$

Implementation of this formula requires reconstruction of the implicit stream function values at V_0 and V_1 using the piecewise solenoidal velocity field. To this end we denote the two elements that share an edge

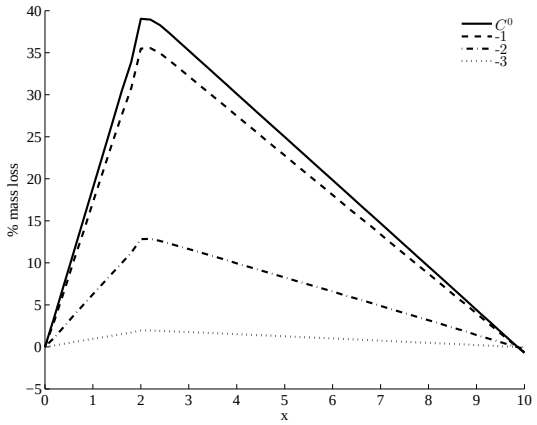


Figure 2.20: *Comparison of the mass loss in the discontinuous velocity LSFEM (2.64) with* $\|\cdot\|_{(h)} = h\|\cdot\|_0$*, and* $\alpha = -1, -2, -3$ *vs. a standard* C^0 *LSFEM (2.23) for the backward step domain on a less refined mesh (1649 triangles).*

$\varepsilon = (\varepsilon_1, \varepsilon_2)$ by $\kappa^+(\varepsilon)$ and $\kappa^-(\varepsilon)$. For a given $\boldsymbol{u}_h \in \mathbf{D}_r(\Omega)$ let ψ_k^+ and ψ_k^- denote its implicit stream functions on each $\kappa^+(\varepsilon)$ and $\kappa^-(\varepsilon)$, respectively:

$$\boldsymbol{u}_h^\pm = (u_{h,1}^\pm, u_{h,2}^\pm) = \boldsymbol{u}_h|_{\kappa^\pm(\varepsilon)} = (\partial_y \psi_k^\pm, -\partial_x \psi_k^\pm). \tag{2.67}$$

Solving for the gradients of the implicit stream functions yields

$$\nabla \psi_k^\pm = (-u_{h,2}^\pm, u_{h,1}^\pm). \tag{2.68}$$

As a result, along edge ε

$$\frac{d\psi_k^\pm}{ds}|_\varepsilon = \nabla \psi_k^\pm \cdot \varepsilon = \big(u_{h,1}^\pm \varepsilon_2 - u_{h,2}^\pm \varepsilon_1\big) = \boldsymbol{u}_h^\pm \times \varepsilon\,. \tag{2.69}$$

The values of the implicit stream functions $\psi_k^\pm$ at V_0 and V_1 can be determined by solving the edge ODEs

$$\begin{cases} \dfrac{d\psi_k^\pm}{ds}|_\varepsilon = \big(\boldsymbol{u}_h^\pm \times \varepsilon\big)|_\varepsilon \\ \psi_k^\pm(0) = C_0 \end{cases} \quad \text{and} \quad \begin{cases} \dfrac{d\psi_k^\pm}{ds}|_\varepsilon = -\big(\boldsymbol{u}_h^\pm \times \varepsilon\big)|_\varepsilon \\ \psi_k^\pm(|\varepsilon|) = C_1 \end{cases} \tag{2.70}$$

for $0 < s < |\varepsilon|$. If the mesh is aligned with the coordinate axes, then closed form solutions are straightforward, while for general unstructured grids we solve (2.70) numerically. For illustration, using the explicit Euler method yields

$$\begin{aligned} \psi_k^\pm(V_0) = \psi_k^\pm(0) &\approx C_1 - |\varepsilon|\big(\boldsymbol{u}_h^\pm(V_0) \times \varepsilon\big) \\ \psi_k^\pm(V_1) = \psi_k^\pm(|\varepsilon|) &\approx C_0 + |\varepsilon|\big(\boldsymbol{u}_h^\pm(V_1) \times \varepsilon\big)\,. \end{aligned} \tag{2.71}$$

Then, using (2.71) in (2.66) gives the approximation

$$\int_{\varepsilon}[\psi_h]^2 d\ell \approx \frac{|\varepsilon|^3}{2}\Big([\big(\boldsymbol{u}_h(V_0)\times\varepsilon\big)]^2+[\big(\boldsymbol{u}_h(V_1)\times\varepsilon\big)]^2\Big)\,. \tag{2.72}$$

Recall that in the dS-VP functional (2.41) we weight the integral of $[\psi_h]^2$ along ε by h^{-3}. To determine the proper weight for the approximation (2.72), observe that dimensional analysis of the terms yields

$$\frac{|\varepsilon|^3}{2}[\big(\boldsymbol{u}_h(V_0)\times\varepsilon\big)]^2+\Big([\big(\boldsymbol{u}_h(V_1)\times\varepsilon\big)]^2\Big)=O(h^3)\quad\text{and}\quad\int_e[\psi_h]^2dl=O(h). \tag{2.73}$$

Therefore, to preserve the relative scaling of the terms in the dS-VP functional (2.41) when the stream function jump is approximated by (2.72) it is necessary to change the weight of this term from h^{-3} to h^{-5}. We add the properly weighted term (2.72) to (2.65) to arrive at the final form of the discontinuous velocity, vorticity, pressure (dV-VP) least-squares functional:

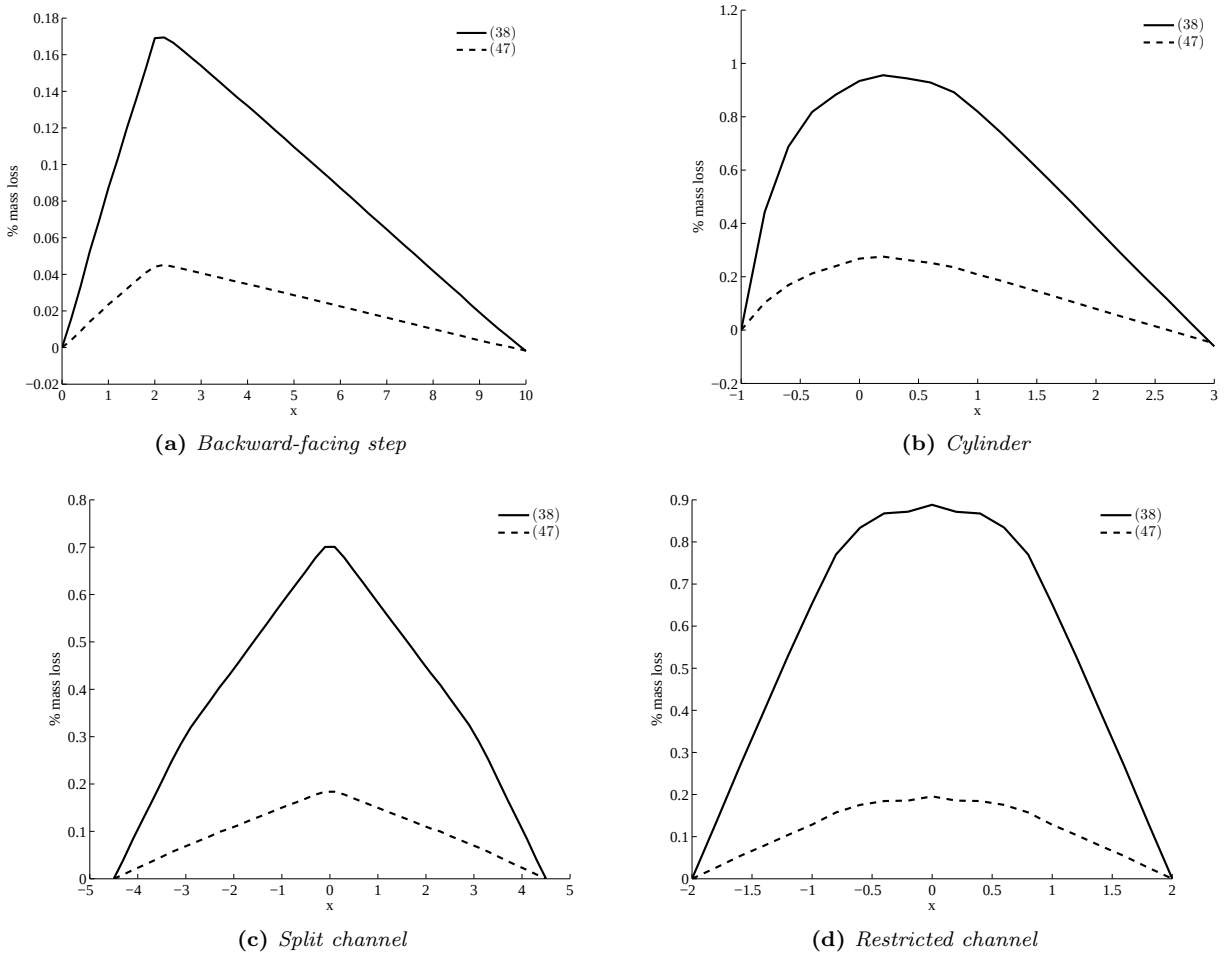
$$\begin{aligned}\widetilde{J}^V_{(h)}(\boldsymbol{u}_h,\omega_h,p_h;\boldsymbol{f}_h)&=\|\nabla\times\omega_h+\nabla p_h-\boldsymbol{f}_h\|^2_{(h)}+\sum_{\kappa\in\mathcal{K}_h(\Omega)}\|\nabla\times\boldsymbol{u}_h-\omega_h\|^2_{0,\kappa}\\&+\sum_{\epsilon\in\mathcal{E}_{h,0}}h^{-3}\|[\boldsymbol{u}_h]\|^2_{0,\epsilon}+h^{-5}\frac{|\varepsilon|^3}{2}\Big([\big(\boldsymbol{u}_h(V_0)\times\varepsilon\big)]^2+[\big(\boldsymbol{u}_h(V_1)\times\varepsilon\big)]^2\Big)\,.\end{aligned} \tag{2.74}$$

To evaluate the role of (2.72) we solve the test problems using both (2.65) and (2.74), implemented with the equal-order space (2.62), and $r=2$. Figure 2.21 shows that inclusion of (2.72) reduces the mass loss from 0.17%, 0.95%, 0.70%, and 0.88% to 0.04%, 0.27%, 0.18%, and 0.13% for each of the test problems respectively. When compared with (2.65), this is a reduction in mass loss by a factor of approximately 4 for each test problem. The improvement in the mass conservation due to (2.72) is even more impressive.

While the piecewise solenoidal fields $\boldsymbol{u}_h\in\mathbf{D}_r(\Omega)$ are curls of discontinuous implicit stream functions $\psi\in[V_{r+1}](\Omega)$, the new dV-VP least-squares method is not equivalent to the dS-VP formulation (2.41), and has some important computational advantages. Because the velocity is approximated directly, implementation of the velocity boundary condition is straightforward for (2.74). Furthermore, for moderate polynomial degrees the dimension of $[V_{r+1}](\Omega)$ is almost twice that of the piecewise solenoidal space $\mathbf{D}_r(\Omega)$.

2.6.4 Preconditioning of the algebraic equations

We denote $\mathbb{K}$ as the symmetric and positive definite matrix resulting from the dV-VP least-squares functional (2.74). For a test function $(\boldsymbol{u}_i,\omega_i,p_i)\in\overline{X}^r_h$, or $(\boldsymbol{u}_i,\omega_i,p_i)\in\overline{X}^{(r)}_h$ we see that the weak form of (2.74) leads



(a) *Backward-facing step* **(b)** *Cylinder*

(c) *Split channel* **(d)** *Restricted channel*

Figure 2.21: *Comparison of the mass loss in the discontinuous velocity LSFEM with* $\|\cdot\|_{(h)} = h\|\cdot\|_0$*, with (2.74) vs. without (2.65) the implicit stream function term.*

to the following 3×3 system for $\mathbb{K}$:

$$\begin{pmatrix} \mathbb{K}_{\boldsymbol{u},\boldsymbol{u}} & \mathbb{K}_{\boldsymbol{u},\omega} & \mathbf{0} \\ \mathbb{K}_{\boldsymbol{u},\omega} & \mathbb{K}_{\omega,\omega} & \mathbb{K}_{\omega,p} \\ \mathbf{0} & \mathbb{K}_{\omega,p} & \mathbb{K}_{p,p} \end{pmatrix} \begin{pmatrix} \vec{\boldsymbol{u}} \\ \vec{\omega} \\ \vec{p} \end{pmatrix} = \begin{pmatrix} f_{\boldsymbol{u}} \\ f_{\omega} \\ f_p \end{pmatrix} \tag{2.75}$$

where

$$\begin{aligned} (\mathbb{K}_{\boldsymbol{u},\boldsymbol{u}})_{ij} = &\sum_k (\nabla \times \boldsymbol{u}_i, \nabla \times \boldsymbol{u}_j)_{0,k} + \sum_{\varepsilon} h^{-3}([\boldsymbol{u}_i],[\boldsymbol{u}_j])_{0,\varepsilon} \\ &+ \sum_{\varepsilon} h^{-5} \frac{|\varepsilon|^3}{2} \Big([\big(\boldsymbol{u}_i(V_1) \times \varepsilon\big)][\big(\boldsymbol{u}_j(V_1) \times \varepsilon\big)] + [\big(\boldsymbol{u}_i(V_0) \times \varepsilon\big)][\big(\boldsymbol{u}_j(V_0) \times \varepsilon\big)] \Big), \end{aligned} \tag{2.76}$$

and

$$(\mathbb{K}_{\boldsymbol{u},\omega})_{ij} = (\nabla \times \boldsymbol{u}_i, \omega_j), \tag{2.77a}$$

$$(\mathbb{K}_{\omega,\omega})_{ij} = h^2(\nabla \times \omega_i, \nabla \times \omega_j), \tag{2.77b}$$

$$(\mathbb{K}_{\omega,p})_{ij} = h^2(\nabla \times \omega_i, \nabla p_j) = h^2(\boldsymbol{n} \times \omega, \nabla p)_{0,\Gamma}, \tag{2.77c}$$

$$(\mathbb{K}_{p,p})_{ij} = h^2(\nabla p_i, \nabla p_j). \tag{2.77d}$$

The h^2 weights arise from the use of the mesh-dependent norm $\|\cdot\|_{(h)} = h\|\cdot\|_0$. Dimensional analysis of the blocks in $\mathbb{K}$ suggests the approximation

$$\mathbb{K} \sim \widetilde{\mathbb{K}} = \begin{pmatrix} h^{-2}\mathbb{M}_{\boldsymbol{u},\boldsymbol{u}} & h\mathbb{D}_{\boldsymbol{u},\omega} & \mathbf{0} \\ h\mathbb{D}^T_{\boldsymbol{u},\omega} & h^2\mathbb{M}_{\omega,\omega} & h^2\mathbb{M}_{\Gamma} \\ \mathbf{0} & h^2\mathbb{M}^T_{\Gamma} & h^2\mathbb{M}_{p,p} \end{pmatrix}, \tag{2.78}$$

where $\mathbb{M}_{\boldsymbol{u},\boldsymbol{u}}$, $\mathbb{M}_{\omega,\omega}$, and $\mathbb{M}_{p,p}$ are unscaled mass matrices, $\mathbb{M}_{\Gamma}$ is the unscaled "boundary" mass matrix acting only on boundary degrees of freedom, and $\mathbb{D}_{\boldsymbol{u},\omega}$ is unscaled "difference" matrix. The structure of $\widetilde{\mathbb{K}}$ indicates that reduction of its condition number may be possible by balancing the equations through the diagonal preconditioner

$$\mathbb{D}_p = \begin{pmatrix} h^p\mathbb{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbb{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbb{I} \end{pmatrix}, \tag{2.79}$$

where p is a suitable parameter. Figure 2.22 shows numerical estimate of the condition number of $\mathbb{D}_p^{1/2}\widetilde{\mathbb{K}}\mathbb{D}_p^{1/2}$ as function of p. The smallest condition number is achieved when $p = 3$. Our computational studies confirm that this value also extends to $\mathbb{K}$, and thus the preconditioned system becomes

$$\mathbb{K}_{prec} = \mathbb{D}_3^{1/2}\mathbb{K}\mathbb{D}_3^{1/2}. \tag{2.80}$$

A similar diagonal preconditioner can be used for the dS-VP formulation (2.41) and in this case, we observed similar improvements in condition number.

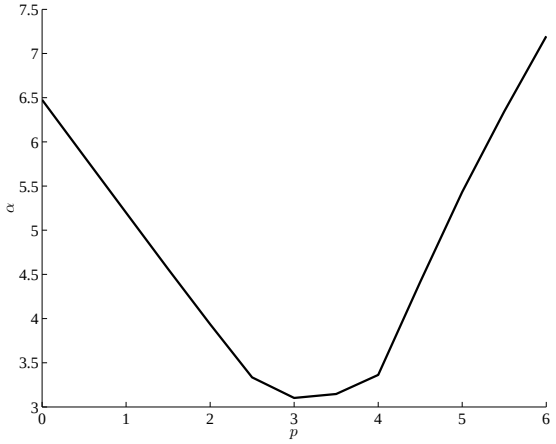


Figure 2.22: *Growth in condition number $O(h^{-\alpha})$ of the preconditioned approximate matrix $\mathbb{D}_p^{1/2}\widetilde{\mathbb{K}}\mathbb{D}_p^{1/2}$ as function of p.*

2.6.5 Computational study

In this section we study the computational properties of the proposed dV-VP least-squares method presented in the previous sections. We implement the method using the equal-order space (2.62) with $r = 2$. Specifically, we study numerically, the convergence rates for the method and the effectiveness of the proposed preconditioner. In addition, velocity profiles for each test problem are plotted for the C^0 formulation (2.18) and the dV-VP formulation (2.74) thus visually demonstrating the improvement in mass conservation.

Convergence

In this section we compare convergence rates of the dV-VP LSFEM with and without the integral jump term. The computational domain Ω is the unit square. $\mathcal{K}_h$ is uniform partition of Ω into square elements with side length equal to $h_i = 2^{-i}$ for $i = 1, 2, 3, 4, 5$. The convergence rates are estimated using a manufactured solution, where the exact solution is selected as

$$\boldsymbol{u} = \begin{bmatrix} -\pi \sin(\pi y) \\ \pi \sin(\pi x) \end{bmatrix}, \quad \omega = \nabla \times \boldsymbol{u} = \pi^2(\cos(\pi x) + \cos(\pi y)), \quad p = \sin(x)\exp(y) \ ,$$

and hence, the corresponding right hand side is

$$\boldsymbol{f} = \begin{bmatrix} -\pi^3 \sin(\pi y) + \cos(x)\exp(y) \\ \pi^3 \sin(\pi x) + \sin(x)\exp(y) \end{bmatrix} .$$

LSFEM	$\widehat{J}^V_{(h)}$ (2.65)				$\widetilde{J}^V_{(h)}$ (2.74)			
h	$\|\boldsymbol{u}-\boldsymbol{u}_h\|_0$	rate	$\|\boldsymbol{u}-\boldsymbol{u}_h\|_1$	rate	$\|\boldsymbol{u}-\boldsymbol{u}_h\|_0$	rate	$\|\boldsymbol{u}-\boldsymbol{u}_h\|_1$	rate
1/4	8.118e-3	–	2.274e-1	–	8.116e-3	–	2.274e-1	–
1/8	1.071e-3	2.922	5.680e-2	2.001	1.071e-3	2.922	5.680e-2	2.001
1/16	1.366e-4	2.947	1.419e-2	2.001	1.366e-4	2.946	1.419e-2	2.001
1/32	1.769e-5	2.950	3.547e-3	2.001	1.769e-5	2.950	3.547e-3	2.001

Table 2.4: *Convergence rates of velocity $\boldsymbol{u}$, for (2.65) and (2.74).*

LSFEM	$\widehat{J}^V_{(h)}$ (2.65)				$\widetilde{J}^V_{(h)}$ (2.74)			
h	$\|\omega-\omega_h\|_0$	rate	$\|\omega-\omega_h\|_1$	rate	$\|\omega-\omega_h\|_0$	rate	$\|\omega-\omega_h\|_1$	rate
1/4	5.040e-2	–	1.007e0	–	5.026e-2	–	1.006e0	–
1/8	4.562e-3	3.466	2.147e-1	2.230	4.563e-3	3.461	2.147e-1	2.228
1/16	5.874e-4	3.211	5.784e-2	2.061	5.876e-4	3.209	5.785e-2	2.061
1/32	1.016e-4	2.982	1.908e-2	1.906	1.016e-4	2.981	1.908e-2	1.905

Table 2.5: *Convergence rates of vorticity ω, for (2.65) and (2.74).*

LSFEM	$\widehat{J}^V_{(h)}$ (2.65)				$\widetilde{J}^V_{(h)}$ (2.74)			
h	$\|p-p_h\|_0$	rate	$\|p-p_h\|_1$	rate	$\|p-p_h\|_0$	rate	$\|p-p_h\|_1$	rate
1/4	8.320e-2	–	7.349e-1	–	8.292e-2	–	7.331e-1	–
1/8	6.525e-3	3.673	1.088e-1	2.756	6.542e-3	3.664	1.089e-1	2.751
1/16	9.049e-4	3.261	2.325e-2	2.491	9.086e-4	3.256	2.327e-2	2.489
1/32	1.922e-4	2.912	5.603e-3	2.333	1.927e-4	2.910	5.609e-3	2.333

Table 2.6: *Convergence rates of pressure p, for (2.65) and (2.74).*

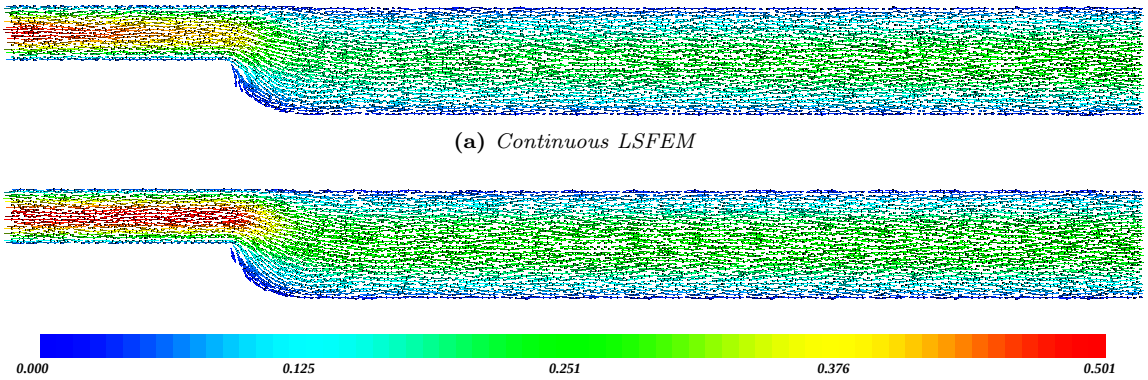
Tables 2.4-2.6 demonstrate that the method indeed exhibits the optimal convergence rates as expected from Theorem 2.3.1. However, since the vorticity and pressure are implemented using quadratic basis functions, we observe that

$$\|\omega-\omega^h\|_0 = \|p-p^h\|_0 = O(h^3) \quad \text{and} \quad \|\omega-\omega^h\|_1 = \|p-p^h\|_1 = O(h^2) \tag{2.81}$$

which is expected for quadratic basis functions. Furthermore, it can be seen that the inclusion of the jump term enforcing the continuity of the implicit stream function, which improved the mass conservation as demonstrated in Section 2.6, does not affect the convergence rates of the method.

Conservation of mass

In Figures 2.23, 2.24, 2.25, and 2.26 the velocity field is plotted for (2.23) and (2.74) with colors representing the magnitude of the vector field. For the backward step, Figure 2.23 shows that the magnitude of the velocity field in the C^0 formulation decreases as the flow reaches the re-entrant corner at $x = 2$ while for (2.74), the initial velocity profile is propagated until the re-entrant corner. For the second test problem, the difference in intensities of the velocity profile at $x = 0$ is clear with a maximum velocity of almost 10.0 in (2.74) compared to only 5.0 for (2.23). In the split channel domain, an initial channel of height 1.0 is split into two channels of height 0.5. Although the height of the two split channels are 0.5, the diameter of the opening is less due to the angle of the split. The velocity profile for (2.74) demonstrates an increase in velocity in the channels with the velocity profile being propagated through the channels. In the C^0 solution, the magnitude of the velocity does not increase relative to the initial velocity and additionally, the magnitude of the velocity dissipates within each of the split channels. The behavior in the restricted channel domain is similar to that of the cylinder flow problem with (2.74) pushing twice as much flow as (2.23) at the narrowest part of the opening.



(a) *Continuous LSFEM*

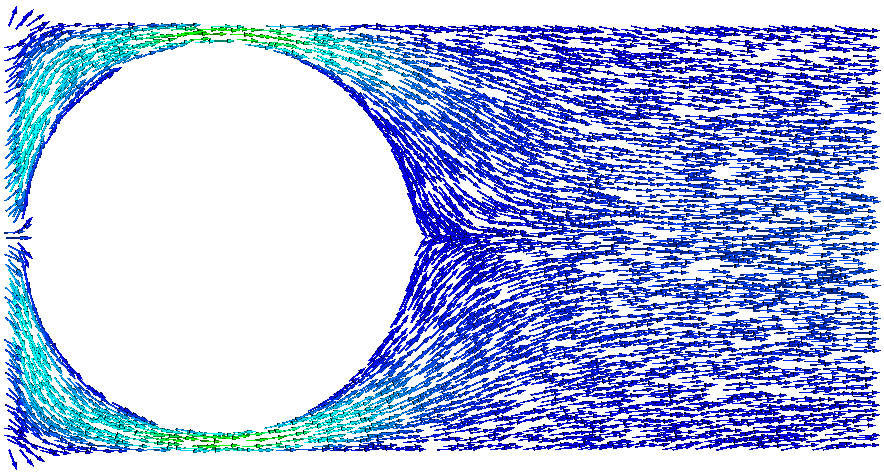
(b) *dV-VP LSFEM*

Figure 2.23: *Velocity plot of (2.18) and (2.74) on the backward step domain.*

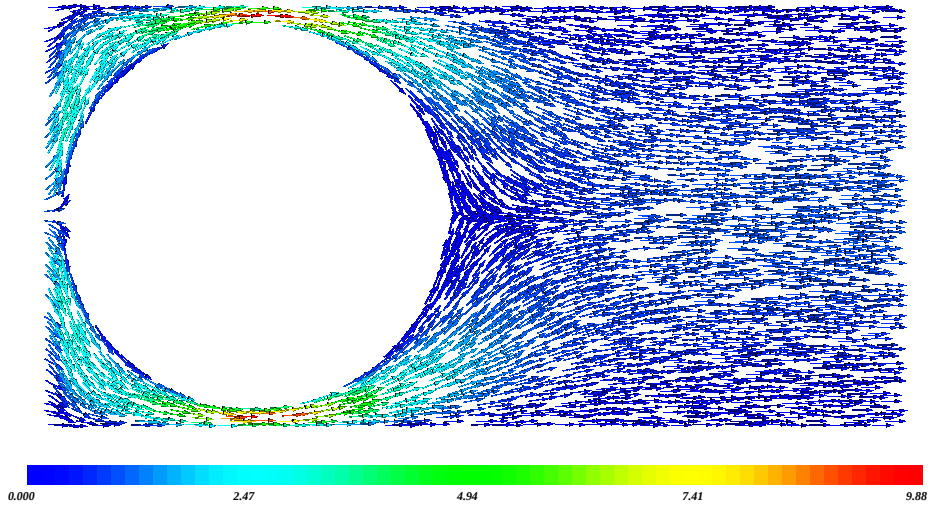
Preconditioning

We next study the effectiveness of the preconditioner in (2.80). We estimate numerically the growth in condition number of the matrix as the mesh is refined for formulations before and after the application of the preconditioners.

Table 2.7 demonstrates that without a preconditioner, the growth in condition number of (2.74) as the mesh is refined is approximately $O(h^{-6})$. The preconditioner (2.80) reduces the growth in the condition number by a factor of 2. In both cases the growth in the condition number is in line with the numerical



(a) *Continuous LSFEM*



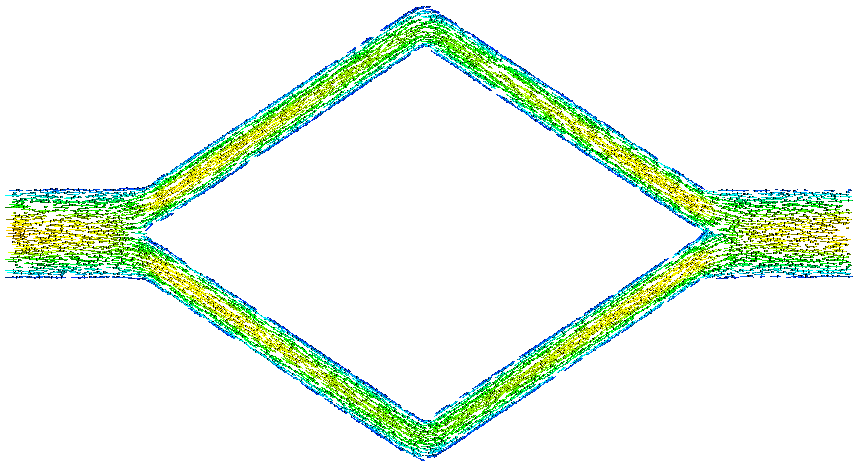
(b) *dV-VP LSFEM*

Figure 2.24: *Velocity plot of (2.18) and (2.74) on the cylinder flow domain.*

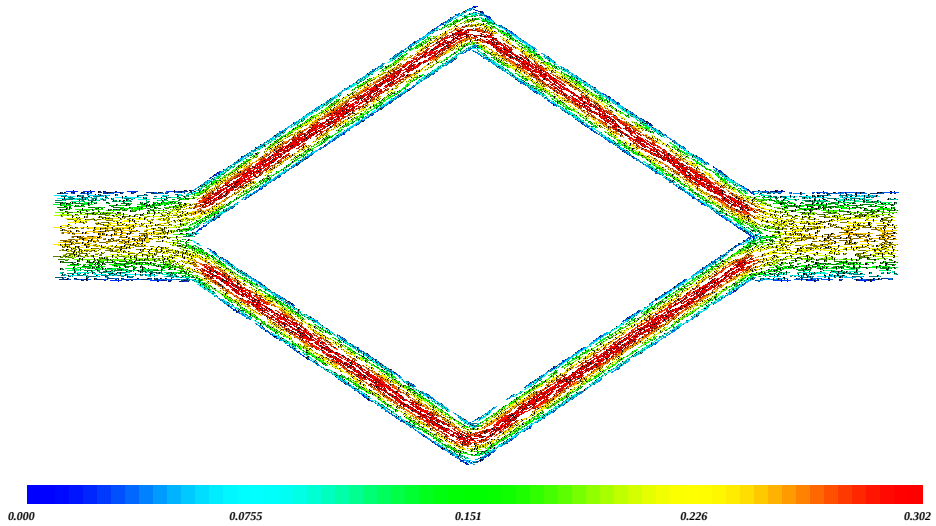
estimates in Figure 2.22. As a point of reference, the dependence on h is $O(h^{-4})$ and $O(h^{-2})$ for (2.23) and (2.30) respectively; see [18, Theorem 4.8, p.119] and [18, Theorem 4.10, p.126]. Therefore preconditioner (2.80) reduces the growth in condition number close to that of the discrete negative norm.

LSFEM	no preconditioning	with preconditioning
$J^V_{(h)}$	3.9	3.9
$\widehat{J}^V_{(h)}$	5.8	2.9
$\widetilde{J}^V_{(h)}$	5.8	2.8

Table 2.7: *Growth in condition number $O(h^{-\alpha})$ for original and preconditioned matrices for (2.37), (2.65), and (2.74).*



(a) *Continuous LSFEM*



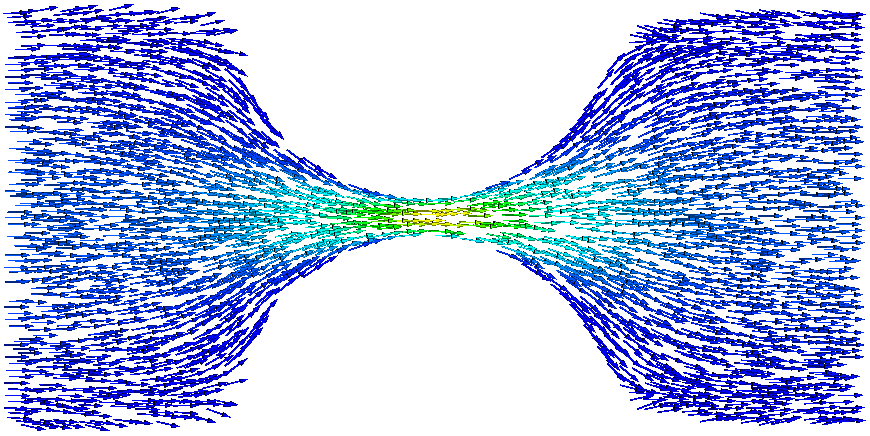
(b) *dV-VP LSFEM*

Figure 2.25: *Velocity plot of (2.18) and (2.74) on the split channel domain.*

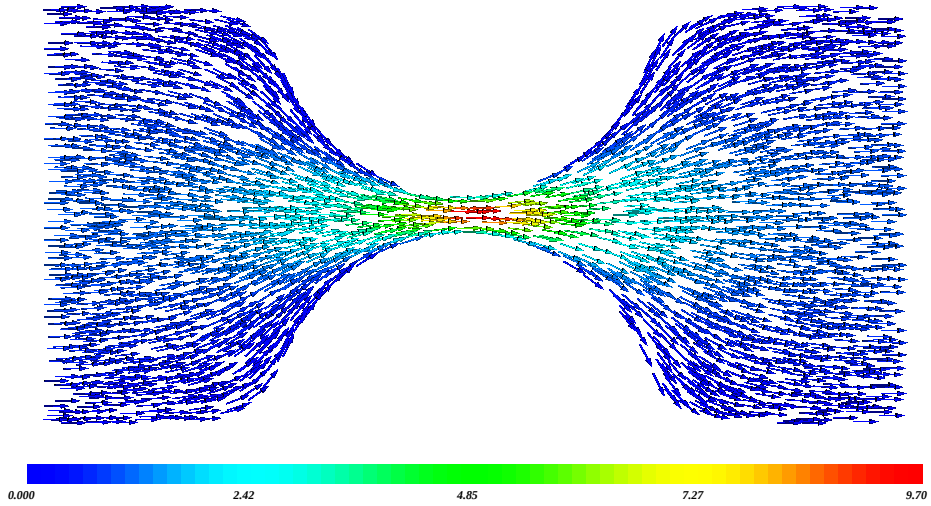
2.7 Navier-Stokes equations

In the previous sections we formulated local mass conserving methods for the Stokes equations. We now apply the methods to the full Navier-Stokes equations. The Navier-Stokes equations differs from the Stokes equations in the momentum equation where there is a nonlinear convective term $(\boldsymbol{u} \cdot \nabla)\boldsymbol{u}$. The equation that governs conservation of mass does not change. Therefore, the methods that improve mass conservation introduced for the Stokes equations generalizes easily to the Navier-Stokes equations.

We develop least-squares methods for the velocity-vorticity-pressure first order form of the Navier-Stokes



(a) *Continuous LSFEM*



(b) *dV-VP LSFEM*

Figure 2.26: *Velocity plot of (2.18) and (2.74) on the restricted channel domain.*

equations. We recall from Section 2.2 the Navier-Stokes equations are given by

$$\begin{cases} -\triangle \boldsymbol{u} + Re(\boldsymbol{u}\cdot\nabla)\boldsymbol{u} + \nabla p = \boldsymbol{f} & \text{on } \Omega \\ \nabla\cdot\boldsymbol{u} = 0 & \text{on } \Omega \end{cases} \tag{2.82}$$

where Re denotes the Reynolds number. The larger the Reynolds number, the more advection-dominated the PDE becomes closer to a hyperbolic type problem.

Following [18] we use the vector identity,

$$(\boldsymbol{u}\cdot\nabla)\boldsymbol{u} = \frac{1}{2}\nabla|\boldsymbol{u}|^2 - \boldsymbol{u}\times\nabla\times\boldsymbol{u} = \frac{1}{2}\nabla|\boldsymbol{u}|^2 + \omega\times\boldsymbol{u} \tag{2.83}$$

and set

$$s = p + \frac{1}{2}|\boldsymbol{u}|^2. \tag{2.84}$$

We can rewrite the convective term in terms of the vorticity and the velocity. The VVP first order form becomes

$$\begin{cases} \nabla \times \omega + Re(\omega \times \boldsymbol{u}) + \nabla s = \boldsymbol{f} & \text{on } \Omega\ , \\ \nabla \times \boldsymbol{u} - \omega = 0 & \text{on } \Omega\ , \\ \nabla \cdot \boldsymbol{u} = 0 & \text{on } \Omega\ . \end{cases} \tag{2.85}$$

The system is completed by the velocity boundary condition (2.11) and zero mean pressure constraint (2.12). Once the solution is found, we can find the original pressure p using the solution for u and s and solving for p in (2.84).

It can be shown that the linearized functional

$$J(\boldsymbol{u}, \omega, s; \boldsymbol{f}) = \|\nabla \times \omega + Re(\omega \times \boldsymbol{u}) + \nabla s - \boldsymbol{f}\|_{-1} + \|\nabla \times \boldsymbol{u} - \omega\|_0 + \|\nabla \cdot \boldsymbol{u}\|_0 \tag{2.86}$$

is norm equivalent on

$$X = \mathbf{H}_0^1(\Omega) \times L^2(\Omega) \times L_0^2(\Omega). \tag{2.87}$$

Therefore, minimization over the finite element subspace

$$X_r^h = \mathbf{V}^r \cap \mathbf{H}_0^1(\Omega) \times V^{r-1} \times V^{r-1} \cap L_0^2(\Omega) \quad r > 1\,. \tag{2.88}$$

gives rise to the following discrete least-squares functional

$$J^h(\boldsymbol{u}^h, \omega^h, s^h; \boldsymbol{f}^h) = \|\nabla \times \omega^h + Re(\omega^h \times \boldsymbol{u}^h) + \nabla s^h - \boldsymbol{f}^h\|_{-h} + \|\nabla \times \boldsymbol{u}^h - \omega^h\|_0 + \|\nabla \cdot \boldsymbol{u}^h\|_0 \tag{2.89}$$

and the minimization problem : *find* $(\boldsymbol{u}^h, \omega^h, s^h) \in X_r^h$ such that

$$J^h(\boldsymbol{u}^h, \omega^h, s^h; \boldsymbol{f}^h) \leq J^h(\boldsymbol{v}, \xi, \omega) \quad \forall (\boldsymbol{v}, \xi, \omega) \in X_r^h \tag{2.90}$$

Approximations of (2.89) with the finite element space X_r^h admit the approximation properties given in the following theorem.

Theorem 2.7.1.

In (2.89) we approximate $\|\cdot\|_{-1}$ using the weighted L^2 approximation; however, similar to the Stokes equations, the more accurate approximation (2.29) can also be used.

Following (2.4), we find that the solution to (2.89) using the following variational problem: *find* $(\boldsymbol{u}^h, \omega^h, s^h) \in X_r^h$ *such that*

$$\begin{aligned} &J((\boldsymbol{u}^h, \omega^h, s^h), (\boldsymbol{v}^h, \xi^h, q^h); f) = \\ &\quad (\nabla \times \omega^h + Re(\omega^h \times \boldsymbol{u}^h) + \nabla s^h - \boldsymbol{f}, \nabla \times \xi^h + Re(\xi^h \times \boldsymbol{u}^h + \omega^h \times \boldsymbol{v}^h) + \nabla q^h)_{(-h)} \\ &\quad +(\nabla \times \boldsymbol{u}^h - \omega^h, \nabla \times \boldsymbol{v}^h - \xi^h)_0 + (\nabla \cdot \boldsymbol{u}^h, \nabla \cdot \boldsymbol{v}^h)_0 = 0 \end{aligned} \tag{2.91}$$

for all $(\boldsymbol{v}^h, \xi^h, q^h \in X_r^h)$. One way to solve the nonlinear PDE (2.91) is using Newton's method.

Similar to Newton's method for root finding, we solve for the zeros of the linearized problem defined by the Jacobian and use the solution to the linearized problem to update the solution. The process is iterated until convergence.

We recall for root finding, Newton's method is given as follows. Let $\boldsymbol{f}$ be a differentiable function, then we iterate

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \mathbf{J}_{\boldsymbol{f}}(\boldsymbol{x}_k)^{-1} \boldsymbol{f}(\boldsymbol{x}_k) \ , \tag{2.92}$$

until convergence, where $\mathbf{J}_{\boldsymbol{f}}(\boldsymbol{x}_k)$ denotes the Jacobian matrix of $\boldsymbol{f}$ evaluated at $\boldsymbol{x}_k$. For simple roots, Newton's method converges quadratically in a neighborhood of the solution. (2.92) can be adapted for finite element methods as well.

The variational problem defined in (2.91) is of the form *find* $\boldsymbol{u} \in X^h$ *such that*

$$F(\boldsymbol{u}, \boldsymbol{v}; \boldsymbol{f}) = \mathbf{0}, \quad \forall \boldsymbol{v} \in X^h. \tag{2.93}$$

Newton's method for (2.93) is given by solving

$$\boldsymbol{u}_{i+1} = \boldsymbol{u}_i + \boldsymbol{s}_i \tag{2.94}$$

where $\boldsymbol{s}_i$ solves the linearized equation

$$\mathbf{J}_F(\boldsymbol{u}, \boldsymbol{v}; \boldsymbol{f}, \boldsymbol{u}_i)\boldsymbol{s}_i = -F(\boldsymbol{u}_i, \boldsymbol{v}; \boldsymbol{f}), \quad \forall \boldsymbol{v} \in X^h. \tag{2.95}$$

In (2.95) $\mathbf{J}_F(\boldsymbol{u}, \boldsymbol{v}; \boldsymbol{f}, \boldsymbol{u})$ is the Jacobian matrix of the functional F with respect to the variables $\boldsymbol{u}$ evaluated

at the previous solution $\boldsymbol{u}_i$.

Applying (2.95) to the nonlinear LSFEM functional for Navier-Stokes (2.91), we find that [18]

$$\begin{aligned}
&\mathbf{J}_F((\boldsymbol{u}^h,\omega^h,s^h),(\boldsymbol{v}^h,\xi^h,q^h);\boldsymbol{f},(\boldsymbol{u}_i^h,\omega_i^h,s_i^h)) = \\
&\quad(\nabla\times\omega_i^h + Re(\omega_i^h\times\boldsymbol{u}_i^h) + \nabla s_i^h - \boldsymbol{f}, Re(\omega^h\times\boldsymbol{v}^h + \xi^h\times\boldsymbol{u}^h))_{(-h)} \\
&\qquad+(\nabla\times\omega^h + Re(\omega_i^h\times\boldsymbol{u}^h + \omega^h\times\boldsymbol{u}_i^h) + \nabla s^h, \nabla\times\xi^h + Re(\omega_i^h\times\boldsymbol{v}^h + \xi^h\times\boldsymbol{u}_i^h) + \nabla q^h)_{(-h)} \\
&\qquad\quad+(\nabla\times\boldsymbol{u}^h - \omega^h, \nabla\times\xi^h - \boldsymbol{v}^h)_0 + (\nabla\cdot\boldsymbol{u}^h, \nabla\cdot\boldsymbol{v}^h)_0
\end{aligned} \tag{2.96}$$

and

$$\begin{aligned}
&F((\boldsymbol{u}_i^h,\omega_i^h,s_i^h),(\boldsymbol{v}^h,\xi^h,q^h);f) = \\
&\quad(\nabla\times\omega_i^h + Re(\omega_i^h\times\boldsymbol{u}_i^h) + \nabla s_i^h - \boldsymbol{f}, \nabla\times\xi^h + Re(\omega_i^h\times\boldsymbol{v}^h + \xi^h\times\boldsymbol{u}_i^h) + \nabla q^h)_{(-h)} \\
&\qquad+(\nabla\times\boldsymbol{u}_i^h - \omega_i^h, \nabla\times\xi^h - \boldsymbol{v}^h)_0 + (\nabla\cdot\boldsymbol{u}_i^h, \nabla\cdot\boldsymbol{v}^h)_0
\end{aligned} \tag{2.97}$$

The Jacobian and functional defined in (2.96) and (2.99) respectively fully define the Newton iteration for the Navier-Stokes functional (2.91). However, in order for Newton's method to converge and converge quadratically, it is necessary to start the iterations in a neighborhood of the solution. This brings us to the problem of finding an initial guess to start the Newton iterations.

Using the fact that for $Re = 0$, the Navier-Stokes equations reduces to the linear Stokes equations, Newton's method converges exactly in one step. It is reasonable to assume that the solutions for close Reynolds numbers are similar. Therefore, for two different Reynolds numbers Re_i, Re_j such that $Re_i - Re_j < \epsilon$ for some $\epsilon \geq 0$, the solution to (2.91) with Re_i is close to the solution of that with Re_j. We can therefore use the solution to (2.91) with Re_j as an initial guess for the Newton iterations for solving (2.91) with Re_i. Such methods for finding solutions to high Reynolds numbers are known as *continuation methods* [18, 57].

Using the same approach as for the dS-VP and dV-VP formulations in Sections 2.5 and 2.6, we replace the velocity space in with the divergence free basis $\mathbf{D}_r$ defined in (2.55). The space that we minimize over therefore is exactly (2.62). Because Newton's method is used to solve the PDE, we apply the divergence free

basis to both the Jacobian and the functional. The Jacobian becomes

$$
\begin{aligned}
&\mathbf{J}_F((\boldsymbol{u}^h,\omega^h,s^h),(\boldsymbol{v}^h,\xi^h,q^h);\boldsymbol{f},(\boldsymbol{u}_i^h,\omega_i^h,s_i^h)) = \\
&\quad \textstyle\sum_{K\in\mathcal{K}}(\nabla\times\omega_i^h + Re(\omega_i^h\times\boldsymbol{u}_i^h) + \nabla s_i^h - \boldsymbol{f}, Re(\omega^h\times\boldsymbol{v}^h + \xi^h\times\boldsymbol{u}^h))_{(-h)} \\
&\qquad + \textstyle\sum_{K\in\mathcal{K}}(\nabla\times\omega^h + Re(\omega_i^h\times\boldsymbol{u}^h + \omega^h\times\boldsymbol{u}_i^h) + \nabla s^h, \nabla\times\xi^h + Re(\omega_i^h\times\boldsymbol{v}^h + \xi^h\times\boldsymbol{u}_i^h) + \nabla q^h)_{(-h)} \\
&\qquad\quad + \textstyle\sum_{K\in\mathcal{K}}(\nabla\times\boldsymbol{u}^h - \omega^h, \nabla\times\boldsymbol{v}^h - \xi^h)_0 + \sum_{\epsilon\in\mathcal{E}_{h,0}} h^{-3}([\boldsymbol{u}^h],[\boldsymbol{v}^h])_{0,\epsilon} + h^{-5}\tfrac{|\varepsilon|^3}{2} \\
&\qquad\qquad + h^{-5}\textstyle\sum_{\epsilon\in\mathcal{E}_{h,0}}\tfrac{|\varepsilon|^3}{2}\Big([\big(\boldsymbol{u}^h(V_0)\times\varepsilon\big)]^2 + [\big(\boldsymbol{u}^h(V_1)\times\varepsilon\big)]^2, [\big(\boldsymbol{v}^h(V_0)\times\varepsilon\big)]^2 + [\big(\boldsymbol{v}^h(V_1)\times\varepsilon\big)]^2\Big)
\end{aligned}
\tag{2.98}
$$

and the functional is

$$
\begin{aligned}
&F((\boldsymbol{u}_i^h,\omega_i^h,s_i^h),(\boldsymbol{v}^h,\xi^h,q^h);f) = \\
&\quad \textstyle\sum_{K\in\mathcal{K}}(\nabla\times\omega_i^h + Re(\omega_i^h\times\boldsymbol{u}_i^h) + \nabla s_i^h - \boldsymbol{f}, \nabla\times\xi^h + Re(\omega_i^h\times\boldsymbol{v}^h + \xi^h\times\boldsymbol{u}_i^h) + \nabla q^h)_{(-h)} \\
&\qquad + \textstyle\sum_{K\in\mathcal{K}}(\nabla\times\boldsymbol{u}_i^h - \omega_i^h, \nabla\times\boldsymbol{v}^h - \xi^h)_0 + \sum_{\epsilon\in\mathcal{E}_{h,0}} h^{-3}([\boldsymbol{u}_i^h],[\boldsymbol{v}^h])_{0,\epsilon} \\
&\qquad\quad + h^{-5}\textstyle\sum_{\epsilon\in\mathcal{E}_{h,0}}\tfrac{|\varepsilon|^3}{2}\Big([\big(\boldsymbol{u}_i^h(V_0)\times\varepsilon\big)]^2 + [\big(\boldsymbol{u}_i^h(V_1)\times\varepsilon\big)]^2, [\big(\boldsymbol{v}^h(V_0)\times\varepsilon\big)]^2 + [\big(\boldsymbol{v}^h(V_1)\times\varepsilon\big)]^2\Big)
\end{aligned}
\tag{2.99}
$$

Here, the notation $\boldsymbol{u}^h(V_0)$ and $\boldsymbol{u}^h(V_1)$ are the same as defined in (2.71) and (2.66).

2.7.1 Computational study

In this section we study some computational aspects of the discontinuous divergence-free Navier-Stokes formulation. To solve the nonlinear PDE, we use Newton's method as described in Section 2.7. The Newton iterations are carried out using the Jacobian from (2.98) and right hand side vector (2.99). The Newton iterations are terminated when the update vector $\boldsymbol{s}_i$ in (2.94) has norm less than 10^{-5}.

Our first study tests the convergence of the method. We test the convergence on the unit square $\Omega = [0,1]\times[0,1]$ discretized by quadrilateral elements of size $h = 2^{-i}$ for $i = 2,3,4,5$. Using the method of manufactured solutions, the exact solution is chosen to be

$$
\boldsymbol{u} = \begin{bmatrix} -\pi\sin(\pi y) \\ \\ \pi\sin(\pi x) \end{bmatrix}, \quad \omega = \nabla\times\boldsymbol{u} = \pi^2(\cos(\pi x) + \cos(\pi y)), \quad s = \sin(x)\exp(y)\ ,
$$

which induces the right hand side vector

$$\boldsymbol{f} = \begin{bmatrix} -\pi^3 \sin(\pi y) - Re\pi^2(\cos(\pi x) + \cos(\pi y))\pi \sin(\pi x) + \cos(x)\exp(y) \\ \\ \pi^3 \sin(\pi x) - Re\pi^2(\cos(\pi x) + \cos(\pi y))\pi \sin(\pi y) + \sin(x)\exp(y) \end{bmatrix}.$$

The first order system is implemented using the equal order space (2.62).

The L^2 and H^1 convergence rates for the velocity u, vorticity ω, and pressure s are shown in Tables 2.8, 2.9, 2.10 respectively.

h	$\|\boldsymbol{u} - \boldsymbol{u}_h\|_0$	rate	$\|\boldsymbol{u} - \boldsymbol{u}_h\|_1$	rate
1/4	6.801e-3	–	2.307e-1	–
1/8	9.581e-4	2.827	5.806e-2	1.992
1/16	1.503e-4	2.750	1.475e-2	1.986
1/32	2.182e-5	2.752	3.791e-3	1.976

Table 2.8: *Convergence rates of velocity $\boldsymbol{u}$, for Navier-Stokes system.*

h	$\|\omega - \omega_h\|_0$	rate	$\|\omega - \omega_h\|_1$	rate
1/4	4.575e-2	–	1.021e0	–
1/8	8.281e-3	2.465	2.822e-1	1.855
1/16	1.594e-3	2.423	7.501e-2	1.883
1/32	2.428e-4	2.505	2.066e-2	1.879

Table 2.9: *Convergence rates of vorticity ω, for Navier-Stokes system.*

h	$\|s - s_h\|_0$	rate	$\|s - s_h\|_1$	rate
1/4	1.968e-1	–	1.156e0	–
1/8	7.424e-2	1.406	1.823e-1	2.664
1/16	1.196e-2	2.021	3.574e-2	2.508
1/32	1.727e-3	2.315	7.116e-3	2.438

Table 2.10: *Convergence rates of pressure s, for Navier-Stokes system.*

Because we implement the equal order space, the tables show that the the H^1 convergence rates are approximately h^2 for each variable which is optimal for quadratic elements.

We now turn to two two commonly used tests problems for the Navier-Stokes equations, the lid-driven cavity and the backward facing step used in the Stokes calculations (Figure 2.1).

For the lid-driven cavity problem, the domain is a unit square with a horizontal velocity on the top wall. The lid-driven cavity domain is shown in Figure 2.27 and is meshed by 6694 triangles. The boundary

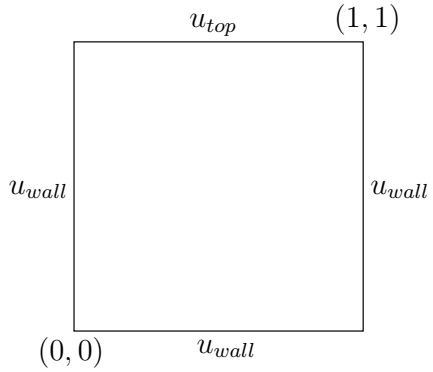


Figure 2.27: *Domain for lid-driven cavity.*

conditions for the lid-driven cavity domain are given by

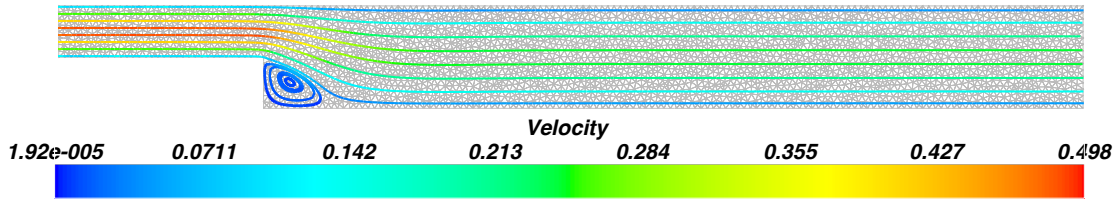
$$\boldsymbol{u}_{top} = \begin{bmatrix} 1 \\ \\ 0 \end{bmatrix} \ , \ \boldsymbol{u}_{wall} = \begin{bmatrix} 0 \\ \\ 0 \end{bmatrix} \tag{2.100}$$

Backward-facing step

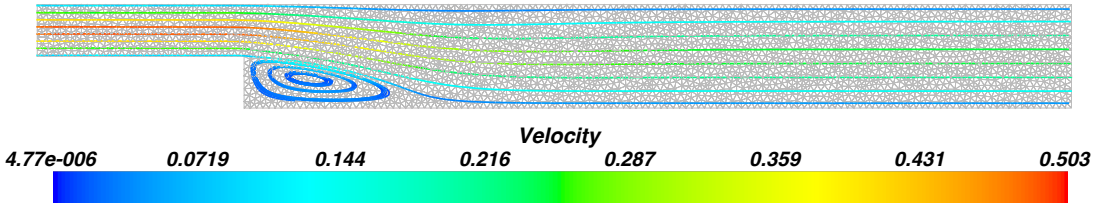
Our first test problem for the Navier-Stokes equations revists the backward-facing step used to test the dV-VP formulation for the Stokes equations. We implement (2.98) with right hand side (2.99) for the domain and boundary conditions described in Figure 2.1 and (2.31) and we solve the equations with Reynolds numbers $Re = 100$ and $Re = 400$. The streamlines for each case are plotted in Figure 2.28. For $Re = 100$ we see a small vortex forming at the corner of the step interface which is not evident in the Stokes equations. As the Reynolds number increases to $Re = 400$, we see a larger vortex forming and the magnitude of the velocity velocity above the vortex is much larger when compared to $Re = 100$ and the standard Stokes equations. The mass loss throughout the domain is summarized in Figure 2.29. It is clear that the dV-VP method performs extremely well with respect to mass conservation with only 0.11% and 0.05% mass loss for $Re = 100$ and $Re = 400$ respectively.

Lid-driven cavity

Upon visual inspection, the streamlines plotted in Figure 2.30 closely match the results in [36]. For $Re = 100$, the central vortex is in the upper right with two small vortices forming in the bottom left and right corners.



(a) $Re = 100$



(b) $Re = 400$

Figure 2.28: *Streamlines for backward-facing step problem with* $Re = 100$ *(a) and* $Re = 400$ *(b). Color gradient shows magnitude of velocity.*

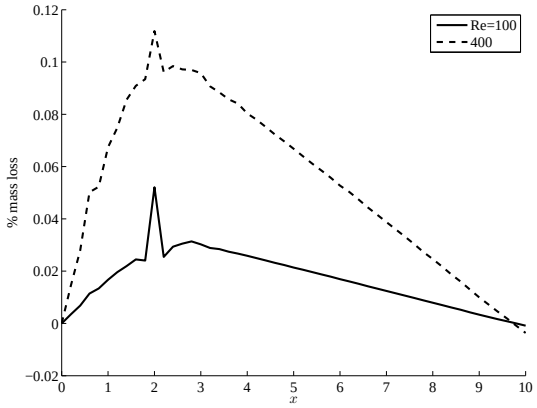
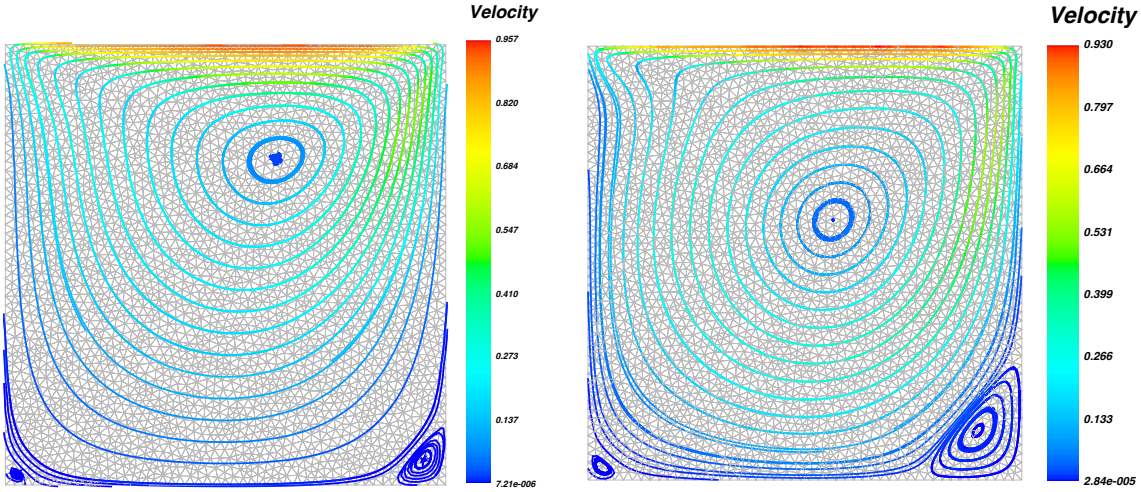


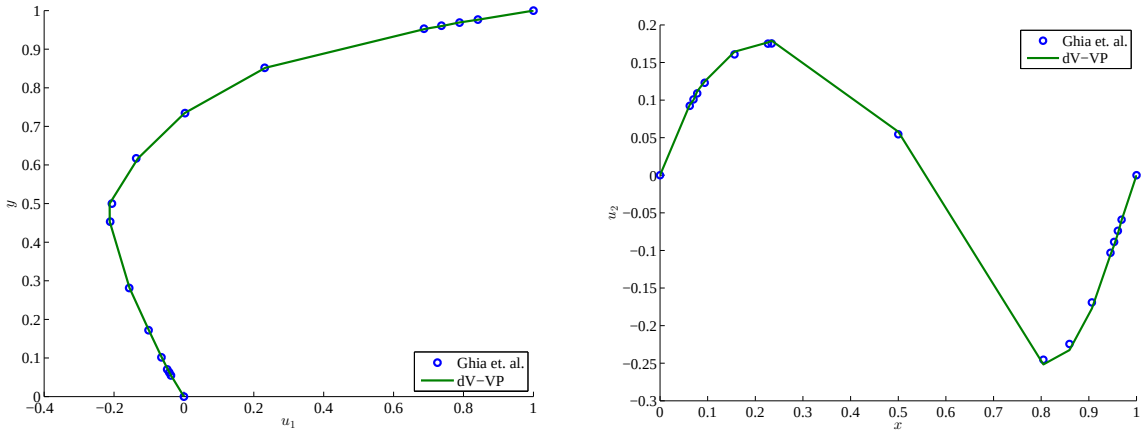
Figure 2.29: *Mass loss for backward-facing step domain for* $Re = 100$ *and* $Re = 400$.

For $Re = 400$, the central vortex is shifted towards the center and the bottom vortices are larger. A more quantitative comparison is shown in Figures 2.31 and 2.32 where the values of the velocities are compared on lines through the center of the domain. For Reynolds number $Re = 100$, the dV-VP method recovers the solution of [36] almost exactly. Furthermore, for $Re = 400$, the dV-VP method performs extremely well considering the mesh size is $h \approx 0.017$. Compared with the C^0 LSFEM results found in [18], the dV-VP method dramatically improves least-squares methods for the Navier-Stokes equations.



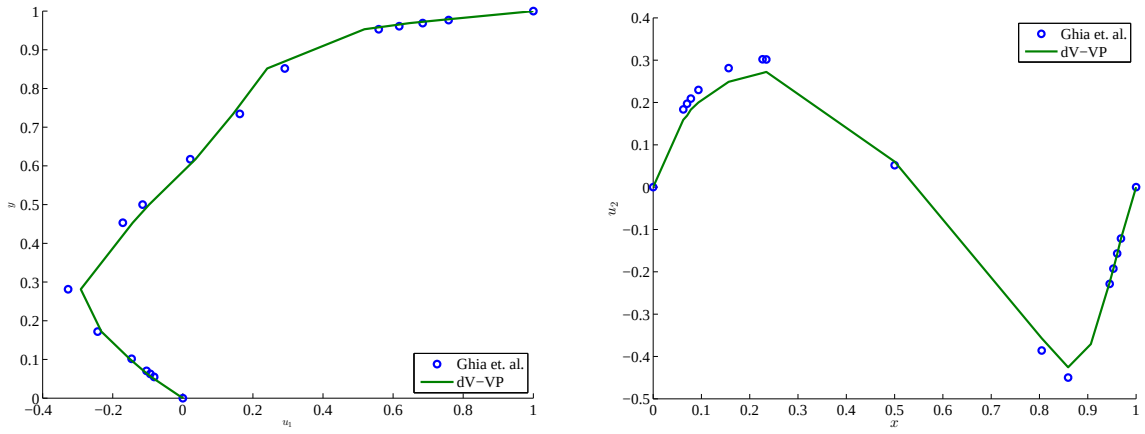
(a) $Re = 100$ **(b)** $Re = 400$

Figure 2.30: *Streamlines for lid-driven cavity with Reynolds number* $Re = 100$ *(a) and* $Re = 400$ *(b). Color gradient shows magnitude of velocity.*



(a) u_1 *on vertical line through center of domain* **(b)** u_2 *on horizontal line through center of domain*

Figure 2.31: *Comparison of velocities vs [36] for Reynolds number* $Re = 100$.



(a) u_1 *on vertical line through center of domain*

(b) u_2 *on horizontal line through center of domain*

Figure 2.32: *Comparison of velocities vs [36] for Reynolds number* $Re = 400$.

Chapter 3

Multigrid methods

This chapter is devoted to efficient iterative methods that solve matricies arising from finite element discretizations. Because of the local nature of standard basis functions, resulting matrices have sparse structure, usually a constant number of nonzeros per row. Hence the number of nonzeros in a matrix is linearly proportional to the number of unknowns. Multigrid is a class of iterative methods that solves linear systems with complexity linear in the number of unknowns. The two main aspects of multigrid are the relaxation and coarse-grid correction. The former is also known as while the later is also known as interpolation.

Relaxation methods such as Jacobi and Gauss-Seidel are extremely efficient in eliminating high-frequency or oscillatory errors. However, after few iterations, the error modes that remain are smooth and of low frequency. As such, the convergence of relaxation methods stagnates. The second aspect of multigrid methods is the coarse-grid correction. The remaining smooth error on the fine-grid is projected onto a coarse-grid using appropriate restriction operators. When projected onto the coarse-grid, the smooth error is again of high-frequency – efficiently solved by relaxation methods. The process is continued recursively until the coarse problem is small enough to solve exactly. To complete the multigrid cycle, the solution to the coarse-grid problem is interpolated to the fine-grid and solution to the fine-grid problem is updated. A successful multigrid method employs relaxation methods that target all high-frequency error modes. The interpolation between coarse and fine grids must complement relaxation in that error not attenuated by relaxation must be in the range of the interpolation operator.

When standard multigrid is run for M-matrices, the methods perform extremely well and solve systems with cost linear in the number of unknowns. However, for PDE discretizations that do not result in M-matrices, standard multigrid methods do not perform optimally and require modifications to either the smoother or the interpolation or both. Because multigrid methods are optimal solvers for H^1 elliptic problems, there has been much interest in expanding the effectiveness of multigrid to new problems.

In recent years there has been significant effort in designing efficient solvers for high-order H^1 discretizations and also lowest-order $H(curl)$ discretizations (edge elements). However, efficient solvers for high-order $H(curl)$ discretizations is lacking. The goal of this thesis is to extend the efficiency of multigrid methods to

high-order discretizations for $H(curl)$ conforming spaces.

Edge elements are often seen in electromagnetic applications where curl conforming elements are needed (e.g. Maxwell's equations). However, due to the nontrivial nullspace of the curl operator, standard iterative methods perform poorly on such problems. The works of [1, 41] present efficient geometric multigrid methods for $H(curl)$ conforming discretizations by introducing specialized smoothers that target the large nullspace of the curl operator. Algebraic methods that generalize the previous geometric methods were introduced in [20, 42]. Such methods induce edge aggregation operators using a corresponding nodal aggregation.

High-order finite elements are often used in practice due to their improved convergence properties, however, their usage presents a number of difficulties for the linear solver. The sparsity of the matrices diminishes as the order of approximation increases. This is natural due to the increase in number and connectivity of the degrees of freedom. The decrease in sparsity of the matrix adversely affects both direct and iterative solvers. Furthermore, the condition number increases with the order of approximation leading to less accurate solutions for direct methods and decreased convergence rates for iterative methods. Because high-order methods come with a number of disadvantages for the linear solver, there is much research in their efficient solutions. [39, 52] formulate multigrid methods for high-order H^1 discretizations. The idea is to use a refined low-order mesh as a preconditioner to the high-order problem. Because multigrid solves low-order discretizations effectively, the inverse of the preconditioner can be found efficiently.

In this chapter we explore multigrid methods that solve high-order $H(curl)$ conforming finite element discretizations. We detail the standard multigrid method in Section 3.3.2, in Section 3.1.1 review existing multigrid methods that solve high order H^1 discretizations, in Section 3.1.2 we review multigrid for the lowest order $H(curl)$ elements. Because high-order finite element bases come in many different varieties, for example interpolatory or hierarchical, we formulate multigrid methods for both types of bases. In Section 3.2 we discuss multigrid methods for hierarchical high-order basis functions [45] and in Section 3.3.1 we develop multigrid for interpolatory bases.

3.1 Multigrid

Multigrid methods originally targeted matrices arising from discretizations of elliptic PDEs. The methods have $O(n)$ cost where n is the number of unknowns and hence are optimal solvers. Multigrid relies on the fact that we are given a matrix that discretizes an elliptic PDE and is optimal for H^1 elliptic problems discretized by C^0 linear finite elements or finite difference methods result in elliptic problems. Relaxation methods such as Gauss-Seidel and Jacobi are extremely effective at eliminating high-frequency error modes.

These are the error modes in the direction of the eigenvector corresponding to the large eigenvalues. However, after few iterations, the convergence of such methods stagnates as only smooth error modes remain. The basic insight that multigrid builds upon is that smooth error when restricted to a coarse grid, is mapped to high-frequency modes on the coarse grid. As such, relaxation methods perform well at eliminating those error modes. Thus, the two main operators are the smoother (relaxation method) and the interpolation operator (mapping between fine and coarse grids). The simplest multigrid algorithm is the V-cycle outlined in Algorithm 1.

Algorithm 1: V_cycle (A, x, b, l)

input : A - matrix, x - initial guess, b - right hand side, l - level
output: x - solution
if $l == maxlevel$
    solve $Ax = b$ using direct method
$x \leftarrow \texttt{relax}(A, x, b)$
$r_f = b - Ax$
$r_c = Rr_f$
$A_c = RAP$
$e_c = 0$
V_cycle $(A_c, e_c, r_c, l + 1)$
$x \leftarrow x + Pe_c$
$x \leftarrow \texttt{relax}(A, x, b)$

Because the general idea of multigrid algorithms is embodied in the simple V-cycle, we utilize Algorithm 1 as a basis for the multigrid methods in this dissertation. Other more complex cycling methods include W-cycles and full multigrid cycling [58] which can improve convergence rates.

3.1.1 Multigrid for high-order H^1

Algebraic multigrid methods have been extended to effectively solve spectral and high-order interpolatory H^1 finite elements problems on quadrilaterals [39] and later for triangles [52] and tetrahedra. In this section we assume that A_p is the discretization of an H^1 elliptic problem using a degree p H^1 conforming basis V^p. Figure 3.1 shows an example of the locations for the $p = 5$ basis functions on a triangle. The basis is then defined as the Lagrange polynomials through those points. The high-order basis satisfies the following approximation property [25]:

Theorem 3.1.1. *Let $\mathcal{K}$ be a quasi-uniform partition of Ω into finite elements. Then for every $u \in H^{p+1}(\Omega)$ there exists a constant $C > 0$ such that*

$$\|u - \Pi(u)\|_0 \leq Ch^{p+1}\|u\|_{p+1} \tag{3.1}$$

where Π *is the finite element interpolation of piecewise polynomials of degree* p.

According to Theorem 3.1.1, the rate of convergence is proportional to the order of approximation used. Therefore, it is possible to obtain extremely accurate results on a moderately refined mesh. However, A_p does not exhibit the desirable properties (M-matrix) for multigrid methods to converge optimally.

The main idea in preconditioning A_p is approximating the high-order basis with a low-order basis on a more refined mesh. The idea is similar to the fact that high-order polynomials can be accurately approximated by piecewise linear polynomials. Conveniently, the high-order degrees of freedom are defined on nodes which induce a Delaunay triangulation shown in Figure 3.1. That is, given a triangular element κ and its high-order basis nodes, denote the Delaunay triangulation of the element κ. Then a new mesh for Ω can be defined as

$$\hat{\mathcal{K}} = \bigcup_{\kappa \in \mathcal{K}} \kappa \tag{3.2}$$

Let M_p denote the resulting matrix from discretizing using V^1 on $\hat{\mathcal{K}}$. In the case for H^1, the number of degrees of freedom in M_p is equal to that in A_p. It can be shown that M_p is an efficient preconditioner for A_p. It is shown in [53] that for Chebyshev spectral methods on tensor grids

$$\|M_p^{-1} A_p\| \leq \frac{\pi^2}{4}. \tag{3.3}$$

Furthermore, because M_p is an M-matrix and H^1 elliptic, its inverse can be computed efficiently using multigrid. M_p is an accurate approximation to the spectrum of A_p and is easily solved, thus, it is an efficient preconditioner for A_p.

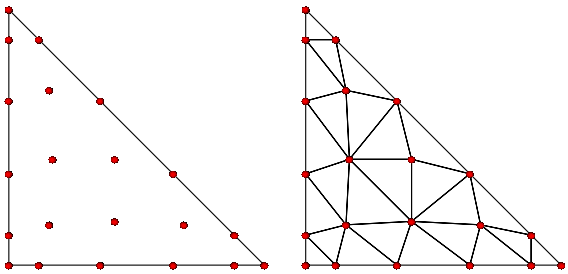


Figure 3.1: *Locations of* H^1 *basis functions of order* $p = 5$ *(left) and the Delaunay triangulation of the degree of freedom points (right).*

3.1.2 Multigrid for lowest-order $H(curl)$

The $H(curl)$ function space is composed of vector functions whose components lie in L^2 and their curls also lie in L^2. For smooth domains, the function space $\mathbf{H}^1$ where each component is in H^1 is a proper subset of $H(curl)$. Therefore, using $\mathbf{H}^1$ to approximate problems with solutions in $H(curl)$ may result in incorrect solutions and/or unphysical behavior. Another vector function space is $H(div)$ which contains elements in which each componenent lies in L^2 and the divergence lies in L^2. Formally,

$$\begin{aligned} H(curl,\Omega) &= \{\boldsymbol{u} \in \mathbf{L}^2(\Omega) \mid \nabla \times \boldsymbol{u} \in \mathbf{L}^2(\Omega)\} \ , \\ H(div,\Omega) &= \{\boldsymbol{u} \in \mathbf{L}^2(\Omega) \mid \nabla \cdot \boldsymbol{u} \in L^2(\Omega)\} \ . \end{aligned} \tag{3.4}$$

Furthermore, $\mathbf{H}^1$ is a proper subset of the intersection of $H(curl)$ and $H(div)$. That is,

$$\mathbf{H}^1 \subset H(curl) \cap H(div). \tag{3.5}$$

The equality holds if the domain satisfies certain regularity requirements.

Conforming finite elements for $H(curl)$ were introduced by Nédélec in [50, 51]. These are vector finite elements that enforce tangential continuity across element boundaries as opposed to C^0 continuity for H^1 basis functions. Due to the tangential continuity requirement, the basis functions are associated with edges of the element as opposed to nodes for the H^1 element. Because of this, the $H(curl)$ conforming element is also known as the *edge* element and the terminology will be used interchangeably.

Conforming finite elements for $H(div)$ were introduced by Raviart-Thomas in [55]. Similar to the $H(curl)$ conforming element, the $H(div)$ conforming element is a vector finite element, however, the Raviart-Thomas element enforces normal continuity across element boundaries. The normal component lies on the *face* of the element and hence are known as face elements.

Both Nédélec and Raviart-Thomas elements are widely used in engineering applications. For example, Maxwell's equations from electromagnetics often include PDEs that require functions to be in $H(curl)$. Therefore, edge elements are very popular in electromagnetics. Raviart-Thomas elements are often used in mixed finite element methods, for example, Poissons equation where $-\nabla \cdot \nabla u = f$ is often written as a system

$$\begin{cases} -\nabla \cdot \phi = f \\ \nabla u - \phi = 0 \end{cases} \tag{3.6}$$

where $\phi \in H(div)$ and $u \in H^1$. Therefore, both types of vector finite elements are widely employed in science and engineering.

In order to accurately solve differential equations, the vector function spaces $H(curl)$ and $H(div)$ are often used. However, the resulting discrete matrix no longer satisfies the M-matrix properties for multigrid to have optimal convergence rates. Therefore, extending multigrid methods to efficiently solve such systems have been the topic of recent resesarch.

The works of Hiptmair [41] and Arnold et. al. [1] introduce geometric multigrid methods for vector finite elements. Because these methods define appropriate smoothers, they lay out the basis for many multigrid methods to come. An *algebraic* multigrid method was introduced by Reitzinger and Schoberl in [56] where the geometric concept of a mesh is not assumed but deduced through the nonzero structure of a provided auxiliary matrix. The interpolation operator proposed in [56] is piecewise constant. In numerical studies, it was shown to be insufficient to provide h-independent convergence for the multigrid method. The interpolation operator was improved by using ideas from smoothed aggregation [62] where the constant interpolation operator is smoothed in order to more accurately capture the low energy modes [20]. Furthermore, an even more accurate interpolation operator where near h-independence is achieve was proposed in [42]. Our multigrid methods for high-order $H(curl)$ finite elements builds upon and extends the methods for lowest order edge elements and we therefore detail those algorithms in this section.

Governing equations and discretization

The target equations we are interested in are in the form of the eddy current problem from electromagnetics. The equations are given by

$$\nabla \times \nabla \times \boldsymbol{u} + \sigma \boldsymbol{u} = \boldsymbol{f} \tag{3.7}$$

for $\sigma \geq 0$ with either tangential Dirichlet boundary conditions

$$\boldsymbol{n} \times \boldsymbol{u} = 0 \tag{3.8}$$

or Neumann conditions

$$\boldsymbol{n} \times \nabla \times \boldsymbol{u} = 0. \tag{3.9}$$

The corresponding weak problem is given by: *find* $\boldsymbol{u} \in H(curl)$ *such that*

$$\underbrace{(\nabla \times \boldsymbol{u}, \nabla \times \boldsymbol{v})_0}_{S} + \sigma \underbrace{(\boldsymbol{u}, \boldsymbol{v})_0}_{M} = (\boldsymbol{f}, \boldsymbol{v})_0 \tag{3.10}$$

for all $\boldsymbol{v} \in H(curl)$.

For discretizations of (3.10) with $H(curl)$ conforming finite elements, S is the stiffness matrix and M is the mass matrix. We denote the discrete system as

$$A = S + \sigma M. \tag{3.11}$$

As σ increases, the relative importance of the curl-curl term decreases and the problem becomes easier to solve. The difficulty for multigrid in solving A with small σ is due to the nullspace of the stiffness matrix S which has a nullspace that requires special treatment in both the smoother and the interpolation operator. Differing from the Laplacian, which has null space consisting of only a single mode, the constant function, the curl-curl operator has an infinite dimensional nullspace consisting of gradients of scalar functions. This follows from the vector calculus identity

$$\nabla \times \nabla \phi = 0 \tag{3.12}$$

for any differentiable ϕ. For $H(curl)$ conforming discretizations, such as the Nédélec element, (3.12) holds on the discrete level. To understand this, consider the de Rham complex

$$H^1 \xrightarrow{\nabla} H(curl) \xrightarrow{\nabla\times} H(div) \xrightarrow{\nabla\cdot} L^2 \tag{3.13}$$

where the image of one mapping is equal to the kernel of the next. That is,

$$\nabla \times (\nabla \phi) = 0 \tag{3.14}$$

$$\nabla \cdot (\nabla \times \boldsymbol{u}) = 0 \tag{3.15}$$

(3.14) is known as the exact sequence property. For $H(curl)$ and $H(div)$ conforming finite elements (3.13) holds on a discrete level, that is

$$V^r \xrightarrow{\nabla_h} \mathbf{Q}^r \xrightarrow{\nabla\times_h} \mathbf{F}^r \xrightarrow{\nabla\cdot_h} N^r \tag{3.16}$$

where $V^r \subset H^1, \mathbf{Q}^r \subset H(curl), \mathbf{F}^r \subset H(div), N^r \subset L^2$ are the nodal, edge, face, and volume element spaces respectively. The operators $\nabla_h, \nabla\times_h, \nabla\cdot_h$ denote discrete versions of the gradient, curl, and divergence operators. Assuming lowest order elements, the discrete operators $\nabla_h, \nabla\times_h, \nabla\cdot_h$ are also defined using mesh connectivity information. The usage of such operators to define PDEs is the subject of discrete exterior calculus [2].

For the lowest-order case, the discrete gradient operator is simple and can be obtained as follows. We

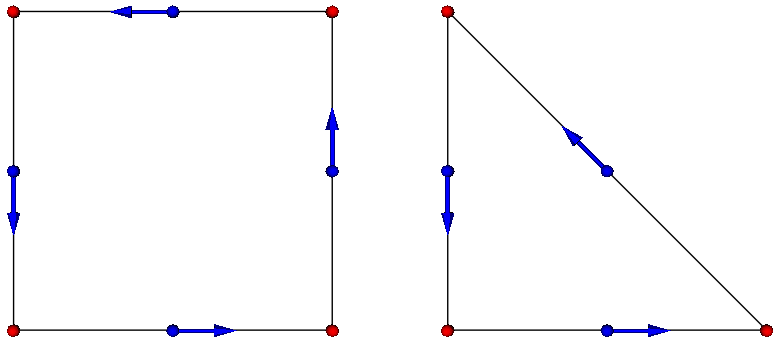


Figure 3.2: *Edge element basis function locations on quadrilateral and triangle. Basis functions have unit tangent on associated edge and vanishing tangent on all other edges.*

now denote the discrete gradient operator

$$\mathbb{D}_1 : V^1 \to \mathbf{Q}^1 \tag{3.17}$$

which maps between the nodal element space V^1 and the edge element space $\mathbf{Q}^1$. Each row represents an edge and has two nonzeros. Now suppose $e_i = (v_j, v_k)$, then row i contains a -1 in column j and a 1 in column k. Nodal basis functions are normalized to unity at the node they are associated with and vanish linearly to zero to adjacent nodes. Edge basis functions have unit tangent component for the edge they are associated with and hence, the action of the discrete gradient operator sets the value of an edge as the signed difference between the vertices or nodes that it connects. Thus, elements of the edge element space that are differences of nodal values represent gradient fields and hence are the nullspace modes of the curl operator.

Because of the exact sequence property, the nullspace of the curl stiffness matrix contains gradients of all functions in V^r. Therefore, the size of the nullspace for S is equal to $|V^r|$. For the lowest order case, the elements of V^1 correspond to the nodes of the mesh and the elements of $\mathbf{Q}^1$ correspond to the edges of the mesh. The number of nullspace modes for S is equal to the number of nodes in the mesh.

The lowest-order Nédélec basis is $H(curl)$ conforming and can be used to discretize (3.10). It is common to show the degrees of freedom on a finite element diagram shown in Figure 3.2 The basis has the property that the *tangent component has unit tangent on one edge with vanishing tangent component on all other edges* and can be characterized as

$$\mathbf{Q}^1 = \{a + \boldsymbol{b} \times \boldsymbol{x} \mid a \in \mathbb{R}, \boldsymbol{b} \in \mathbb{R}^2\} \tag{3.18}$$

However, (3.18) does not provide much insight into the shape and behavior of the basis functions. More

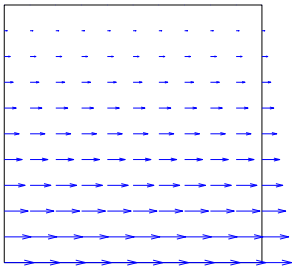


Figure 3.3: *Nédélec basis on a quadrilateral for bottom edge. Vector field has tangent equal to* $(1,0)^T$ *on bottom edge. Tangent vanishes on all other edges.*

specifically, on quadrilaterals, $\mathbf{Q}^1$ can be written as

$$\mathbf{Q}^1 = \begin{bmatrix} P_0(x)P_1(y) \\ \\ P_1(x)P_0(y). \end{bmatrix} \tag{3.19}$$

where P_r denotes a polynomial of degree r. For the reference quadrilateral $(0,1)\times(0,1)$, the basis functions are

$$\boldsymbol{q}_0(x,y) = \begin{bmatrix} 1-y \\ \\ 0 \end{bmatrix}, \quad \boldsymbol{q}_1(x,y) = \begin{bmatrix} 0 \\ \\ x \end{bmatrix}, \quad \boldsymbol{q}_2(x,y) = \begin{bmatrix} -y \\ \\ 0 \end{bmatrix}, \quad \boldsymbol{q}_3(x,y) = \begin{bmatrix} 0 \\ \\ x-1 \end{bmatrix}. \tag{3.20}$$

The basis can be seen in Figure 3.3. For each basis function, the tangent component is unity along a single edge and vanishes on on other edges.

$$\boldsymbol{q}_i(x,y) = \lambda_i \nabla\lambda_{(i+1)} - \lambda_{(i+1)}\nabla\lambda_i, \quad i = 0,1,2. \tag{3.21}$$

where λ_i denotes the barycentric coordinates associated with node i and for $i = 2$ we let $i+1 = 0$. On a unit triangle with nodes $\{(0,0),(1,0),(0,1)\}$ the barycentric coordinates are

$$\begin{aligned} \lambda_0(x,y) &= 1-x-y \\ \lambda_1(x,y) &= x \\ \lambda_2(x,y) &= y \end{aligned} \tag{3.22}$$

The triangle edge element basis is visualized in Figure 3.4 and although the vector field is more interesting than the quadrilateral basis, it still satisfies the edge element property.

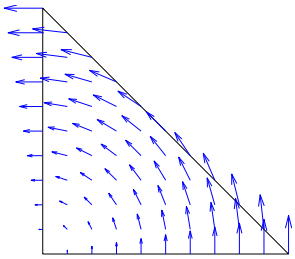


Figure 3.4: *Nédélec basis on a triangle for diagonal edge. Vector field has tangent equal to* $(-1,1)^T$ *on diagonal edge. Tangent vanishes on left and bottom edges.*

3.1.3 Multigrid methods

In order for multigrid methods to be successful, special care needs to be taken to handle the nullspace of S. We first review geometric multigrid methods which design smoothers for curl type problems. In this dissertation, we use the method of Hiptmair [41]; however, one can also use the methods of Arnold et. al. [1].

The Helmholtz decomposition of a vector field $\boldsymbol{u} \in H(curl)$ is given by

$$\boldsymbol{u} = \nabla\phi + \nabla \times \boldsymbol{v} \tag{3.23}$$

for $\phi \in H^1$ and $\boldsymbol{v} \in H(curl)$. Thus, elements of $H(curl)$ have two components that are inherently different and need to be treated separately. Because the stiffness matrix S contains only the curl part of the Helmholtz decomposition, pointwise relaxation schemes do not attenuate oscillatory error modes that are images of gradient functions. It is this observation that that leads us to define *hybrid* smoothers in which highly oscillatory modes in both spaces.

Because the eddy current matrix A from (3.10), is of the form $S+\sigma M$, where σ is nonzeros, the gradient modes in the image of A are nonzero. The idea of a hybrid smoother is to smooth both the image of the curls and the image of the gradients. Standard relaxation schemes tackle only the image of the curl as those are the modes that are present in the image of A. We can project onto the gradient space using the discrete gradient operator $\mathbb{D}_1$ defined in (3.17) by performing

$$\mathbb{D}_1^T A \mathbb{D}_1. \tag{3.24}$$

To see the action of (3.24), consider

$$\begin{aligned} \mathbb{D}_1^T A \mathbb{D}_1 &= \mathbb{D}_1^T (S + \sigma M) \mathbb{D}_1 && (3.25) \\ &= \mathbb{D}_1^T S \mathbb{D}_1 + \sigma \mathbb{D}_1^T M \mathbb{D}_1 && (3.26) \\ &= 0 + \sigma \mathbb{D}_1^T M \mathbb{D}_1 && (3.27) \end{aligned}$$

The first term is 0 due to the exact sequence property and the second term is a Laplacian like matrix. Therefore, the gradient space is non trivial in the image of A. The algorithm for hybrid smoothing is detailed in Algorithm 2.

Algorithm 2: hybrid_smooth$(A, x, b, \mathbb{D})$

input : A - matrix, x - initial guess, b - right hand side, $\mathbb{D}$ - discrete gradient
output: x - solution
$x \leftarrow \texttt{relax}(A, x, b)$
$r \leftarrow b - Ax$
$\hat{x} \leftarrow \texttt{relax}(\mathbb{D}^T A \mathbb{D}, 0, \mathbb{D}^T r)$
$x \leftarrow x + \mathbb{D}\hat{x}$
$x \leftarrow \texttt{relax}(A, x, b)$

Hybrid smoothing first performs one iteration of relaxation on the image of A (edge element space), followed by one iteration of iteration on the gradient space (nodal space) using (3.24). Once the smoother is defined, a geometric multigrid method is automatic as the interpolation between grid levels is simply linear finite element interpolation. Algebraic approaches obtain coarse grids without rediscretizing the problem. We now recall the works of [56], and [20] that explore algebraic multigrid methods for edge element discretizations.

Algebraic multigrid methods do not rediscretize the problem but form coarse problems using algebraically determined interpolation and restriction operators. Multigrid is known to perform well for H^1 elliptic problems. Therefore, the idea is to use a multigrid hierarchy for a nodal discretization to induce an edge hierarchy. In order to do so, the the method requires an auxiliary matrix given in which linear H^1 conforming finite elements are used to discretize the following PDE associated with (3.10): *find* $u \in V^1$ *such that*

$$(\nabla u, \nabla v)_0 + \sigma (u, v)_0 \tag{3.28}$$

for all $v \in V^1$. We denote the discretized matrix for (3.28) as B. Because the degrees of freedom for the linear nodal basis functions correspond with the nodes of the mesh, the nonzero structure of B provides us with the connectivity of the mesh. Nodes are diagonal entries and any off diagonal entries are edge

connections with other nodes. Thus, if $a_{ij} \neq 0$, then there exists an edge in the mesh from vertex i to vertex j. Therefore, by examining the sparsity structure of B, we can obtain the discrete gradient operator $\mathbb{D}_1$.

The method of [56] utilizes a nodal multigrid hierarchy for B to induce an edge multigrid hierarchy. The idea is to define the edge interpolation operators to make the de Rham diagram of Figure 3.5 commute.

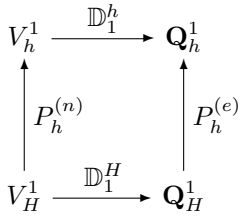


Figure 3.5: *De Rham diagram for lowest order.*

V_h^1 and V_H^1 denote the fine and coarse nodal spaces respectively while $\mathbf{Q}_h^1$ and $\mathbf{Q}_H^1$ denote the fine and coarse edge spaces respectively. The fine and coarse discrete gradient operators are denoted by $\mathbb{D}_1^h$ and $\mathbb{D}_1^H$ respectively. Here, the nodal prologation operator is denoted by $P_h^{(n)}$ and the edge prolongation operator by $P_h^{(e)}$. We assume that $P_h^{(n)}$ is obtained in the setup phase of multigrid on B and $\mathbb{D}_1^h$ either given or constructed using the nonzero structure of B. Therefore, it is necessary to define $\mathbb{D}_1^h$ and $P_h^{(e)}$ so that Figure 3.5 commutes, i.e.,

$$P_h^{(e)} \mathbb{D}_1^H = \mathbb{D}_1^h P_h^{(n)}. \tag{3.29}$$

In defining $P_h^{(e)}$ and $\mathbb{D}_H^1$ in this way the order does not matter. There are two options:

1. first perform a coarse discrete gradient and then interpolate in the edge element space
2. first interpolate in the nodal element space and then perform a fine discrete gradient.

The commuting diagram property ensures that the nullspace for the coarse grid consists of gradient modes. Thus, $\mathbb{D}_1^H$ forms a basis for the nullspace of the coarse grid curl stiffness matrix.

Edge interpolation $P_h^{(e)}$ follows from $P_h^{(n)}$ by noticing that nodal aggregates induce edge aggregates. Denote by $\mathcal{A}_j$ the set of nodes in aggregate j. Then a coarse edge is defined as an edge between two nodal aggregates (see Fig. 3.6), and a coarse edge exists only if a fine edge exists that connects two nodes of separate aggregates. From this description of edge aggregation, we define interpolation $P_h^{(e)}$ from coarse edges to fine edges.

Formally, assuming $i = (i_1, i_2)$ is a fine edge and $j = (j_1, j_2)$ is a coarse edge, then the edge interpolation

operator is defined as

$$P_h^{(e)}(i,j) = \begin{cases} 1, & \text{if } i_1 \in \mathcal{A}_{j_1} \text{ and } i_2 \in \mathcal{A}_{j_2} \ , \\ -1, & \text{if } i_2 \in \mathcal{A}_{j_1} \text{ and } i_1 \in \mathcal{A}_{j_2} \ , \\ 0, & \text{otherwise} \ . \end{cases} \tag{3.30}$$

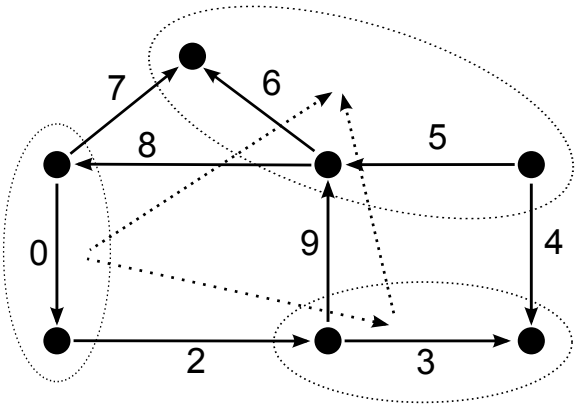


Figure 3.6: *Example of $P_h^{(e)}$ operator. Nodal aggregates are contained within the dotted ellipses. The induced edges are represented by dotted edges.*

For example, in Figure 3.6, the nodal aggregates are represented by dotted ellipses and the induced coarse grid edges are dotted lines. In this example, the edge interpolation operator is

$$P^{(e)} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.31}$$

Once $P_h^{(e)}$ is defined, a coarse discrete gradient $\mathbb{D}_1^H$ is automatic since

$$\mathbb{D}_1^H = \left(\left(P_h^{(e)} \right)^T P_h^{(e)} \right)^{-1} \left(P_h^{(e)} \right)^T \mathbb{D}_1^h P_h^{(n)}. \tag{3.32}$$

Because $P_h^{(e)}$ has full rank, and each fine edge belongs in exactly one aggregate, $(P_h^{(e)})^T P_h^{(e)}$ is a diagonal matrix and its inverse is easily computed.

The $P_h^{(e)}$ defined in (3.30) are piecewise constant. Although defining $P_h^{(e)}$ in this way ensures the commutativity of the diagram, it does not yield an accurate interpolation operator. As a result, it has been demonstrated through numerical studies that the convergence of the resulting method decreases as the mesh is refined. The convergence can be improved by adapting ideas from smoothed aggregation [20] in which prolongation smoothers are designed to smooth tentative prolongation operators in order to more accurately capture low energy modes. Thus, one iteration of weighted Jacobi smoothing on $P_h^{(e)}$ has been shown to be effective. The smoothed interpolation operator becomes

$$\hat{P}_h^{(e)} = \mathcal{S} P_h^{(e)} \tag{3.33}$$

where $\mathcal{S}$ is the matrix form of weighted Jacobi

$$\mathcal{S} = I - \alpha D^{-1} A \tag{3.34}$$

where $D = \text{diag}(A)$ and $\alpha = \dfrac{4}{3\rho(D^{-1}A)}$ where $\rho(\cdot)$ is the spectral radius. By doing so, the resulting method becomes less dependent on h.

In a following paper [42], showed a further improved method with respect to h independence and in the process drops the requirement of the auxiliary matrix B. The insight is that when using the tentative *nodal* prolongation operator $P_h^{(n)}$ is piecewise constant and hence the nodal hierarchy does not produce h independent multigrid convergence. Thus, if instead we use a *smoothed* nodal prolongation operator to induce an edge prolongation operator then the resulting method should have improved convergence rates.

If S and M are separated, then one obtains the improved prolongation operator via

$$\breve{P}_h^{(e)} = \left(I - \alpha D_S^{-1} S + \beta \mathbb{D}_1^h D_M^{-1} \mathbb{D}^T M \right) P_h^{(e)} \tag{3.35}$$

where $D_S = \text{diag}(S)$ and $D_M = \text{diag}(\mathbb{D}^T M \mathbb{D})$. The constants are defined as $\alpha = \dfrac{4}{3\rho(D_S^{-1}S)}$ and $\beta = \dfrac{4}{3\rho(D_M^{-1}\mathbb{D}^T M \mathbb{D})}$. Here, the first term $I - \alpha D_S^{-1} S$ represents smoothing the edge prolongation operator and

the second term $\beta \mathbb{D}_1^h D_M^{-1} \mathbb{D}^T M$ represents smoothing of the nodal prolongation operator.

3.2 Multigrid for high-order hierarchical $H(curl)$

We now turn our attention to efficient iterative solvers for high-order $H(curl)$ discretizations. High-order bases come in many flavors, and one important type is the hierarchical basis function where the spaces are nested so that

$$\mathbf{Q}^1 \subset \mathbf{Q}^2 \subset ... \subset \mathbf{Q}^p. \tag{3.36}$$

Thus for any $q < p$, $\mathbf{Q}^q \subset \mathbf{Q}^p$. Hierarchical basis functions are useful when p refinement is necessary and is a common basis of choice when implementing *hp*-finite element methods.

First we define the hierarchical basis in Section 3.2.1. In Section 3.2.2 we define multigrid method for hierarchical basis wherein we develop high-order discrete gradient operators for the hierarchical basis and interpolation operators to fully define the multigrid hierarchy. The methods developed in this section are detail in [45].

3.2.1 Hierarchical basis

A hierarchical basis for $H(curl)$ was first defined in [64] and later in [59, 65] to satisfy the de Rham complex. The basis is composed of three types of basis functions: the lowest order edge functions, high-order edge functions, and high-order interior functions. The basis satisfies the de Rham complex by building from a hierarchical basis for H^1. To this end, the gradients of the H^1 basis are used to construct the $H(curl)$ basis, and thus naturally satisfy the complex. Since $\{u \,\|\, u = \nabla\phi,\, \phi \in H^1\} \subsetneq H(curl)$, it is necessary to enrich the $H(curl)$ basis with additional, linearly independent functions. A benefit of this particular $H(curl)$ basis is that the gradients of the H^1 basis are directly represented, thus leading to a straightforward description of the (weak) kernel of $\nabla\times$. In the following, we define basis functions according to the reference element shown in Fig. 3.7.

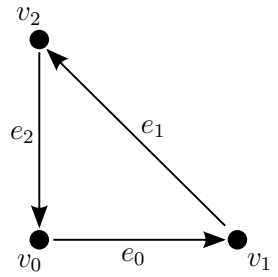


Figure 3.7: *Reference element numbering of nodes and edges.*

In contrast to H^1 elements where the lowest order is $p = 1$–i.e., the linear elements—the lowest order $H(curl)$ conforming elements are with $p = 0$. For example, given a triangle (v_0, v_1, v_2), the lowest order $H(curl)$ basis function associated with edge $e_0 = (v_0, v_1)$ is given by the Whitney form

$$\phi_0(x, y) = \lambda_{v_0} \nabla \lambda_{v_1} - \lambda_{v_1} \nabla \lambda_{v_0}, \tag{3.37}$$

where λ_{v_i} denotes the barycentric coordinates or, in the case of a triangle whose endpoints are $(0, 0), (1, 0), (0, 1)$, the linear H^1 Lagrange basis function for node v_i. We classify the lowest order $H(curl)$ basis functions as $p = 0$ since the Whitney forms have constant tangential component along one edge and vanishing tangential components along all other edges. To define high-order basis functions, it is necessary to introduce Legendre, integrated Legendre, and scaled integrated Legendre polynomials [59, 60].

The Legendre polynomials are a class of orthogonal polynomials, defined on $[-1, 1]$, and are obtained efficiently by the following three term recurrence:

$$l_k(x) = \frac{2k-1}{k} x l_{k-1}(x) - \frac{k-1}{k} l_{k-2}(x), \qquad k \geq 2,$$

with $l_0(x) = 1$, and $l_1(x) = x$. Further, the *integrated* Legendre polynomials are obtained by integrating the Legendre polynomials over $[-1, x]$ so that

$$L_0(x) = \frac{1-x}{2}, \quad L_1(x) = \frac{x+1}{2}, \quad \text{and} \quad L_k(x) = \int_{-1}^{x} l_{k-1}(\xi) d\xi, \quad k \geq 2.$$

Aside from the normalization constant, which plays an important role in conditioning, the integrated Legendre polynomials are simply the Lobatto shape functions. In this paper we utilize the scaled integrated Legendre polynomials [59], defined as

$$L_k^S(s, t) = t^k L_k\left(\frac{s}{t}\right).$$

With these basis functions we complete the definition of the hierarchical basis. We begin with high-order edge basis functions. These functions are designed with a tangential component that is polynomial along one edge and vanishing on the other two edges. For an order p basis, we have for each edge e_i,

$$E_{e_i,j}(x, y) = \nabla L_{j+2}^S(\lambda_{v_0} - \lambda_{v_1}, \lambda_{v_0} + \lambda_{v_1}), \qquad 0 \leq j \leq p-1. \tag{3.38}$$

A notable attribute of this subset of the hierarchical basis is that they are directly the gradients of high-order H^1 edge functions. Let $\mathcal{P}^p(\Omega)$ denote the set of polynomials of degree $\leq p$ over a domain Ω. For a set of

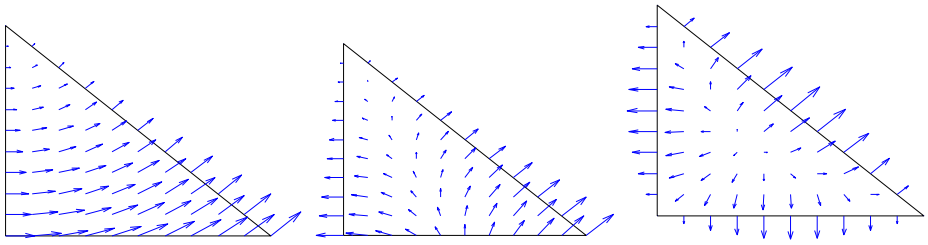


Figure 3.8: *Hierarchical $H(curl)$ basis functions. (left) $p = 0$ basis function (Whitney form) associated with edge e_0. (center) $p = 1$ basis function associated with edge e_0. (right) $p = 2$ interior basis function.*

$p + 1$ edge basis functions, the tangential components span $\mathcal{P}^p(e_i)$. An example of (3.38) is depicted in Figure 3.8.

Next, we define functions that span $\mathcal{P}^p(\tau)$ where τ is the interior of the element. To do this, we use interior bubble functions with vanishing tangential components on the edges. Notice that the functions

$$
\begin{aligned}
u_j(x, y) &= L^S_{j+2}(\lambda_{v_1} - \lambda_{v_0}, \lambda_{v_0} + \lambda_{v_1}), \\
v_k(x, y) &= \lambda_{v_2} l_k(2\lambda_{v_2} - 1) \ ,
\end{aligned}
$$

vanish on edges e_0 and e_2 for u_j, and on edge e_1 for v_k. The result is that the product $u_j v_k$ vanishes on all edges, yielding an interior bubble function. The following three interior basis functions are defined for $0 \leq i + j \leq p - 2$:

$$
\begin{aligned}
F^1_{i,j}(x, y) &= \nabla u_i v_j, \\
F^2_{i,j}(x, y) &= \nabla u_i v_j - \nabla v_j u_i, \\
F^3_j(x, y) &= (\lambda_1 \nabla \lambda_2 - \lambda_2 \nabla \lambda_1) v_j.
\end{aligned}
$$

Here $F^1_{i,j}$ represents the gradient of a corresponding H^1 basis function and $F^2_{i,j}, F^3_j$ are functions that are linearly independent from the rest of the basis. This set of functions forms a basis for a p^{th} order finite element subspace of $H(curl)$ since there are $(p + 1)(p + 2)$ linearly independent functions, which is equal to the size of the space [65]. An example of an interior basis function is depicted in Figure 3.8.

3.2.2 Multigrid method

Because of the hierarchical nature of the basis defined in Section 3.2.1, it is convenient to construct a multigrid hierarchy where coarse grids are low-order approximations of the fine grid, which is reminiscent of p-type multigrid. In order to fully retain a successful AMG method at a coarse level, the kernel of the

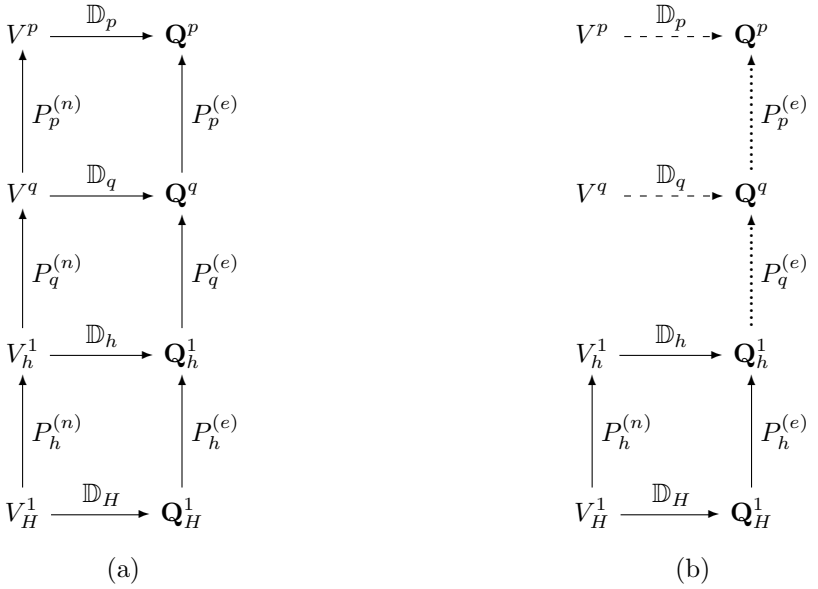


Figure 3.9: *High-order complexes. (a) represents full complex and (b) represents partial complex.*

curl operator should be preserved on the coarse grid. Thus, we extend this idea to high-order levels. For example, consider the diagrams in Figure 3.9.

In Figure 3.9 a, the hierarchy is simply extended from Figure 3.5 to high-order polynomial degrees q and p (with $q < p$). In order to satisfy (3.9a), high-order nodal and edge interpolation operators, along with discrete gradients $\mathbb{D}_q$ and $\mathbb{D}_p$ are required in the multilevel construction. A benefit of the hierarchical construction of the elements is that direct construction of $P_q^{(e)}$ and $P_p^{(e)}$ is straightforward. It is not necessary to aggregate on the auxiliary H^1 problem in order to aggregate in the $H(curl)$ space. Thus, in Fig. 3.9b, nodal interpolation operators are not necessary to induce edge interpolation operators. Moreover, since gradient functions are also inherent in the description, a definition of $\mathbb{D}_q$ and $\mathbb{D}_p$ is possible *without* relating the high-order nodal spaces W_q^0 and W_p^0 to the low-order nodal elements in W_h^0. The high-order discrete gradient operators are necessary to the hybrid smoother, which we define later in this section.

Given an order p discretization of (3.10), we seek a coarse grid corresponding to an order q discretization where $q < p$. The interpolation operator $P_p^{(e)}$ between these levels is straightforward to define. Assuming a convenient ordering of the unknowns, the high-order $P_q^{(e)}$ is defined as

$$P_q^{(e)} = \begin{bmatrix} I_q \\ \\ 0 \end{bmatrix}, \tag{3.39}$$

where I_q is the identity corresponding to the q degrees of freedom.

Because the coarse grid contains all degrees of freedom of a low-order discretization, the kernel of the curl operator is preserved on all levels. Correspondingly, we coarsen the problem until $p = 0$. A central part of our approach is that the low-order AMG algorithm detailed in Section 3.1.2 is applicable to the lowest order level.

To account for the gradient functions, a hybrid Gauss-Seidel[1] smoother on the high-order levels is used. Similar to the low-order case, it is composed of one iteration of Gauss-Seidel followed by one iteration of Gauss-Seidel on the gradient space. But in order to perform relaxation on the gradient space, a high-order discrete gradient operator needs to be defined. Since the hierarchical basis explicitly includes gradients of H^1 functions, it is possible to define a high-order discrete gradient operator. We define the high-order discrete gradient

$$\mathbb{D}_p = \begin{bmatrix} \mathbb{D}_1 & 0 \\ 0 & \hat{\mathbb{D}}_p \end{bmatrix}, \tag{3.40}$$

where $\mathbb{D}_1$ is the lowest order discrete gradient operator for the finest mesh, as defined by (3.17), and $\hat{D}_p$ is a matrix mapping from the high-order H^1 basis to the high-order gradient basis functions in $H(curl)$ defined in the following.

Consider a basis V^p with $V_h^1 \subset V^p$ and identify $\mathcal{N}^0$ as the set of indices associated with basis functions $\phi \in V^p \backslash V_h^1$. Similarly, denote by $\mathcal{N}^1$ the set associated with basis functions $\psi \in \mathbf{Q}^p \backslash \mathbf{Q}_h^1$. Then we define $\hat{\mathbb{D}}_p$, a discrete gradient operator corresponding to high-order degrees of freedom, by identifying basis functions ψ_i for $i \in \mathcal{N}^1$ with $\nabla\phi_j$ for $j \in \mathcal{N}^0$. That is,

$$\hat{D}_p(i,j) = \begin{cases} 1, & \text{if } i \in \mathcal{N}^1, j \in \mathcal{N}^0, \text{ and } \psi_i = \nabla\phi_j, \\ 0, & \text{otherwise .} \end{cases} \tag{3.41}$$

To further motivate this representation of $\mathbb{D}_p$ in (3.40), we consider the first block, which is the lowest order discrete gradient operator $\mathbb{D}_1$. Thus, we preserve the action of the discrete gradient operator for the lowest order degrees of freedom. For the degrees of freedom associated with higher order basis functions, recall from Section 2 that the gradients of H^1 basis functions are explicitly used as basis functions for $H(curl)$. Thus, $\hat{D}_p$ is a mapping between the H^1 basis and their corresponding gradients in the $H(curl)$ space.

As a result of defining $\mathbb{D}_p$ as (3.40), we obtain a high-order discrete gradient operator and a mapping to project $K^{(e)}$ onto the gradient space. The construction of the high-order discrete gradient and high-order

[1]It is possible to use other smoothers such as weighted Jacobi or polynomial smoothers for better parallelizability of the algorithm. For simplicity and consistency with [6, 41, 56] we use Gauss-Seidel.

interpolation operators are detailed in Alg. 3.

Algorithm 3: HO_Hierarchy($\mathcal{G}, levels, p, \mathbb{D}_1$)

for each $q \in levels$
  $\mathbb{D}_q = \begin{bmatrix} \mathbb{D}_1 & 0 \\ 0 & 0 \end{bmatrix}$
  $j = |W_0^0|$
  $k = 0$
  for $i = 0$ **to** $|W_p^1|$ **do**
    if $i \in \mathcal{G}$
      $(\mathbb{D}_q)_{ij} = 1$
      $j = j + 1$
    if $i \in W_q^1$
      $(P_q^{(e)})_{ik} = 1$
      $k = k + 1$
  $\mathcal{G} = \mathcal{G} \setminus \left(\bigcup_{\psi_m \in W_p^1} m \setminus \bigcup_{\psi_n \in W_q^1} n \right)$ {remove high-order d.o.f}
  $p = q$
return $\mathbb{D}_q$, $P_q^{(e)}$ for each $q \in levels$

In this algorithm, let $\mathcal{G}$ denote the set of degrees of freedom that correspond to gradients. Our method utilizing Fig. (3.9b) is detailed in Alg. 4. It is assumed that the high-order discrete gradients D and interpolation operators P are obtained from Alg. 3.

Algorithm 4: HO_AMG($A, x, b, level$)

if $level = maxlevel$
  solve $Ax = b$ using direct method
if $level == HO$
  $x =$ hybrid_smooth($A, x, b, \mathbb{D}_{level}$)
  $r_{fine} = b - Ax$
  $P = P_{level}$ {take HO P as tentative prolongator }
  $r_{coarse} = P^T r_{fine}$
  $A_{coarse} = P^T AP$
  $e_{coarse} = 0$
  HO_AMG($A_{coarse}, e_{coarse}, r_{coarse}$)
  $x = x + Pe_{coarse}$
  $x =$ hybrid_smooth($A, x, b, D[level]$)
else
  SA_EDGE_AMG(A, x, b)

The hybrid smoother used in Algorithm 4 is fully defined with our discrete gradient operator (3.41). Once the high-order discrete gradient operator is obtained, the hybrid smoother (Algorithm 2) is invoked similarly at any order.

An important aspect of multigrid algorithms is $O(n)$ complexity. The amount of work per multigrid

cycle is approximated by the cycle complexity, defined as

$$\mathbb{C}_{cycle} = \frac{\sum_{l=0}^{l_{max}} NNZ(A_l)\nu_l}{NNZ(A_0)} \tag{3.42}$$

where ν_l is the number of smoothing iterations on level l. For the case of $V(1,1)$ cycling, we perform one presmoothing sweep and one postsmoothing sweep–i.e., $\nu_l = 2$ for all l. In order to keep the cycle complexity independent of the order of approximation, each p should not be visited in the hierarchy. Instead, we coarsen an order p discretization by choosing the coarse grid as an order $p/2$ discretization as detailed in the next section.

Computational study

In this section we provide numerical evidence in support of the multilevel approach developed in the previous section. One attractive practical property of the algorithm is the ease of integration into existing smoothed aggregation based codes; we implement our multilevel approach using the PyAMG package [7].

We consider (3.10) on a unit square $\Omega = [0,1] \times [0,1]$ discretized with high-order hierarchical finite elements. In all experiments we use our method to precondition conjugate gradient iterations and, unless otherwise noted, we iterate until the residual has been reduced by 10^8. To obtain the right hand side we take a random vector x and multiply

$$K^{(e)}x = b.$$

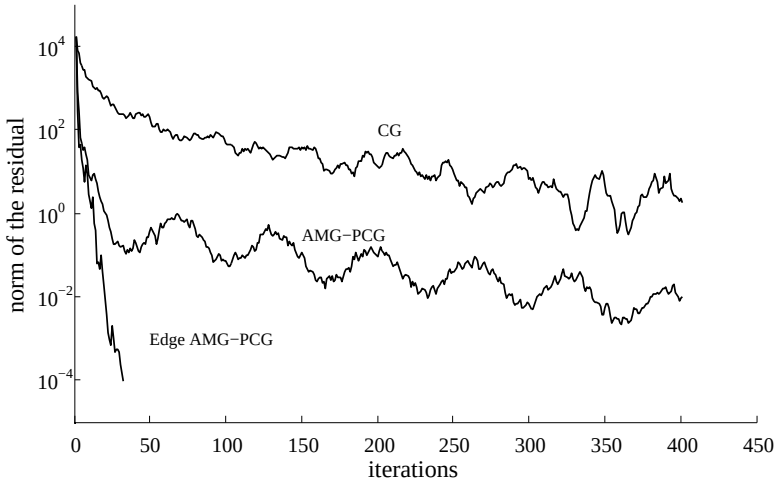


Figure 3.10: *High-order edge AMG PCG compared with conjugate gradient and smoothed aggregation AMG PCG on a $p = 8$ discretization of (3.10) with $\sigma = 10^{-2}$*

First, we compare the residual history of the proposed preconditioner with standard methods in Fig.

3.10. We notice that the residual history of our high-order edge based AMG (labeled "Edge AMG-PCG") indicates an effective preconditioner. In this case, the elements are of very high order ($p = 8$) and the mesh is well refined (500+ elements), yet the residual is more consistently reduced and does not stagnate as in the case of standard AMG, wherein the gradient space is not specifically addressed.

An important component in achieving an efficient reduction of the residual as indicated in Figure 3.10 is the gradient specific smoother. The hybrid smoother specifically targets the gradient space on high-order and low-order levels. The effectiveness of the hybrid smoother is compared with pointwise Gauss-Seidel in Table 3.1. Pointwise Gauss-Seidel drops in performance as the order p increases, due to the enrichment of the near-null space with gradient functions. Hence, as p increases, there are more oscillatory components that pointwise Gauss-Seidel does not attenuate well, leading to a decrease in performance.

$p =$	1	2	3	4	5	6	7	8	9
Hybrid	12	11	11	10	12	15	21	30	46
Pointwise GS	15	14	15	15	20	25	37	53	85

Table 3.1: *Iteration counts for different smoothers on a mesh with* 512 *elements and* $\sigma = 10^{-2}$.

In addition to iteration count, we compare the cost of the hybrid smoother with that of pointwise Gauss-Seidel. Since the hybrid smoother performs smoothing on the gradient space, we compute the additional work performed during hybrid smoothing. The additional work is proportional to the number of nonzeros in the gradient space. Thus, the additional work per multigrid cycle is obtained by summing the number of nonzeros in the gradient space over each level in the hierarchy, i.e.,

$$\mathbb{W}_D = \sum_i \frac{NNZ(D_i^T A_i D_i^T)}{NNZ(A_i)} \times 100. \tag{3.43}$$

The additional work is summarized in Table 3.2.

$p =$	1	2	3	4	5	6	7	8	9
	78.5	62.0	47.0	42.9	38.1	35.7	34.5	34.2	33.4

Table 3.2: *Percent additional work* ($\mathbb{W}_D$) *in smoothing gradient space.*

$p =$	1	2	3	4	5	6	7	8	9
Gradient space	13	12	12	11	15	19	25	38	60
Pointwise GS	15	14	15	15	20	25	37	53	85
Pointwise GS (2 iterations)	14	13	14	14	16	20	28	40	55

Table 3.3: *PCG iteration count for different smoothers*

To compare the work of the hybrid smoother with that of pointwise Gauss-Seidel we use one iteration of *forward* Gauss-Seidel on A followed by one iteration of *symmetric* Gauss-Seidel on the gradient space $D^T AD$ completed by one iteration of *backward* Gauss-Seidel on A. Thus, the additional work of the hybrid smoother will only be a single iteration of symmetric Gauss-Seidel on the gradient space. We compare the effectiveness of smoothing on the gradient space with an additional iteration of smoothing on the entire matrix in Table 3.3. It is evident that the hybrid smoother is more effective than two iterations of pointwise Gauss-Seidel while performing less work.

Although pointwise Gauss-Seidel is not as effective as the hybrid smoother, it still attenuates gradient-like error. This is a result of the chosen basis since the gradient fields are explicitly defined as degrees of freedom. In contrast, when the gradient fields are not explicitly defined, such as in interpolatory bases, pointwise Gauss-Seidel is not as effective.

The hybrid smoother plays an important role in maintaining an efficient solver; moreover, it relies on our choice of the discrete gradient operator to mimic the process described in Fig. (3.9b). For example, since gradient functions are only introduced for high-order elements, it is natural to consider a simpler view of the discrete gradient operator that accounts for only these functions. That is,

$$D_p' = \begin{bmatrix} 0 & 0 \\ \\ 0 & \hat{D}_p \end{bmatrix}.$$

In Table 3.4 we compare this choice with the proposed high-order discrete gradient operator, D_p, in (3.40).

$p =$	1	2	3	4	5	6	7	8	9
D_p	12	11	11	10	12	15	21	30	46
D'_p	14	13	14	14	15	18	24	35	50

Table 3.4: *Iteration count for discrete gradient operators on a mesh with* 512 *elements and* $\sigma = 10^{-2}$.

Although the difference is not severe, D_p consistently outperforms D'_p. By including the D_0 in the $(1,1)$ block of D_p, D_p is a more accurate representation of a high-order discrete gradient operator. The additional cost in using D_p over D'_p is summarized in Table 3.5 by summing the additional number of nonzeros produced in the gradient space over all levels of the hierarchy. The additional cost is computed by comparing the total number of nonzeros present in the gradient space when using the two different discrete gradient operators. For $p < 3$, D_p performs more overall work. However, since the additional work in using D_p diminishes with increasing p, we see that for $p \geq 3$, choosing D_p indeed is more efficient.

$p =$	1	2	3	4	5	6	7	8	9
Additional cost	47.4	24.7	10.7	7.5	4.5	2.6	1.8	1.6	1.2

Table 3.5: *Percent additional cost in using* D_p *over* D'_p *on a mesh with* 512 *elements and* $\sigma = 10^{-2}$.

As $\sigma \to 0$, the problem moves closer to a pure curl-curl problem with a rich gradient near-null space. As a result, conditioning degrades, as depicted in Figure 3.11. The condition number grows as $\sigma \to 0$ and increases strongly with p (exponentially).

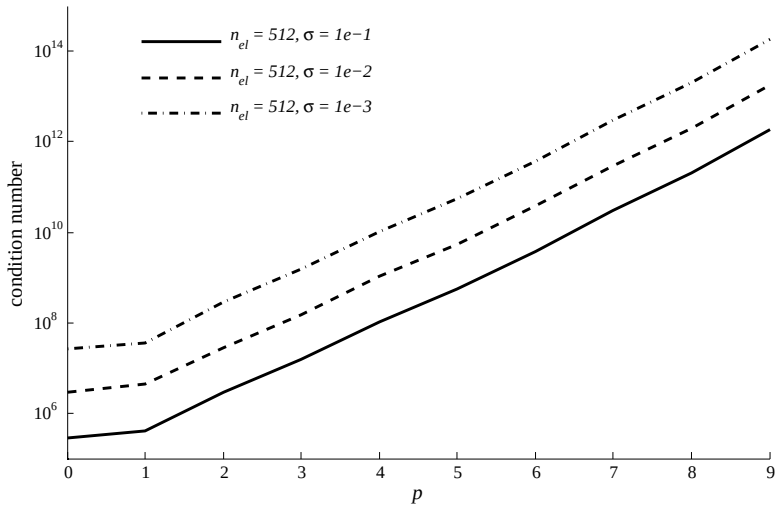


Figure 3.11: *Growth in condition number of $K^{(e)}$ for different σ and number of elements, n_{el}.*

Yet, our method exhibits only modest dependence on p and invariance with respect to typical choices of σ. Indeed, as detailed in Table 3.6, as $\sigma \to 0$, the iterations stay constant (or decrease). Since the contribution of the curl-curl stiffness term becomes more dominant, this indicates that high-order gradient smoothing is effective at targeting the near-null space.

$p =$	1	2	3	4	5	6	7	8	9
$\sigma = 10^{-1}$	13	11	11	10	13	17	24	32	58
10^{-2}	12	11	11	10	12	15	21	30	46
10^{-3}	12	11	11	10	12	13	18	25	40

Table 3.6: *Dependence on σ using 512 elements.*

Our method also exhibits robustness in p as the finite element mesh is refined. Table 3.7 shows a growth in h-refinement consistent with previous low-order results [20, 56]. Even at higher orders, the dependence on the mesh remains the same.

$n_{el} =$	32	128	512	2048	8192
$p = 1$	6 / 6	8 / 9	12 / 16	18 / 26	24 / 44
2	7 / 7	8 / 9	11 / 16	18 / 26	25 / 44
3	8 / 8	8 / 10	11 / 15	18 / 25	25 / 44
4	10 / 10	10 / 10	10 / 16	18 / 26	25 / 44
5	14 / 13	14 / 14	12 / 16	18 / 25	25 / 44

Table 3.7: *Dependence on h with $\sigma = 10^{-2}$ using smoothed / unsmoothed $P_h^{(e)}$ on low-order grids.*

In Table 3.7, we also show the benefit of a *smoothed* prolongation operator $P_h^{(e)}$. As in [20] we observe both accuracy and efficiency gains by using a smoothed prolongation operator on the lowest order grids.

n_{el}	110	440	1760
$p = 1$	7	13	20
2	8	13	20
3	8	13	20
4	10	12	20
5	14	14	20

Table 3.8: *Dependence on h with $\sigma = 10^{-2}$ on unstructured meshes.*

The dependence on h is compared for unstructured meshes in Table 3.8, and the dependence on h is similar to the structured case.

In the previous section, we advocate coarsening in p that reduces the coarse grid to $p/2$, for complexity reasons. In Table 3.9 we compare the convergence and cycle complexity of different coarsening schemes. If we coarsen to just $p - 1$, then we obtain the best convergence rate out of the three methods; however, the cycle complexity increases with p. In contrast, by coarsening directly to $p = 0$, we obtain constant cycle complexity with respect to p; however, the convergence deteriorates in this case, particularly at the highest

orders. Instead, if we coarsen to $p/2$, we find a balance between cycle complexity and convergence.

	$p \to p-1$	$p \to \frac{p}{2}$	$p \to 0$
$p=1$	12 / 2.60	12 / 2.60	12 / 2.60
2	11 / 2.66	11 / 2.66	12 / 2.15
3	10 / 3.03	11 / 2.26	12 / 2.06
4	9 / 3.46	10 / 2.50	12 / 2.03
5	9 / 3.90	12 / 2.27	14 / 2.16
6	12 / 4.31	15 / 2.35	17 / 2.01
7	15 / 4.79	21 / 2.23	24 / 2.01
8	21 / 5.30	30 / 2.36	32 / 2.00
9	29 / 5.71	46 / 2.25	60 / 2.00

Table 3.9: *Iterations and cycle complexity for different coarsening schemes,* 512 *elements,* $\sigma = 10^{-2}$.

3.3 Multigrid for high-order interpolatory $H(curl)$

The second type of high-order basis is interpolatory. Interpolatory bases follow the typical high-order definition wherein degrees of freedom are defined as dual to a unisolvent set of points. Such bases are used widely in finite element packages for example Trilinos/Intrepid [21, 38] and FEniCS [46]. We define the high-order $H(curl)$ conforming basis in Section 3.3.1.

In this section we introduce two multigrid schemes for high-order basis of interpolatory type. Both methods utilize hybrid smoothing and hence require the definition of high-order discrete gradient operators. Due to the nature of high-order bases, the construction of the discrete gradient operator is non-trivial. In Section 3.3.2 we show how to efficiently construct the discrete gradient operator for the interpolatory basis. In our first multigrid method, we use a coarsening scheme inspired by the p-multigrid scheme that was shown to be effective for hierarchical bases. In our second method we adapt the high-order multigrid methods for H^1 defined in Section 3.1.1 to high-order $H(curl)$ bases.

3.3.1 High-order basis

An $H(curl)$ conforming basis requires the tangent component of the vector field to be continuous across element boundaries; therefore, the degrees of freedom are associated with the tangent component of each edge. Thus, the functions in $\mathbf{Q}^p(\hat{\kappa})$ satisfy the following property

$$\boldsymbol{w}_i(\boldsymbol{y}_j) \cdot \tau_j = \delta_{ij}, \qquad \forall i, j \in [1, \dim(\mathbf{Q}^p(\hat{\kappa}))] \ , \tag{3.44}$$

where $\boldsymbol{y}_j$ are nodal locations and τ_j are tangential directions. For interpolatory bases, we have

$$\dim(\mathbf{Q}^p(\hat{\kappa})) = p(p+2). \tag{3.45}$$

The space $\mathbf{Q}^p(\hat{\kappa})$ contains p degrees of freedom per edge and the remaining $p(p-1)$ degrees of freedom are internal. The edge points are chosen to be Gauss points while the internal points are chosen to be the internal points of a degree p Fekete [61] or Warp-Blend [63] set. Similar to the H^1 case, the use of these points reduces the exponential growth in condition number when using equispaced points. For example,

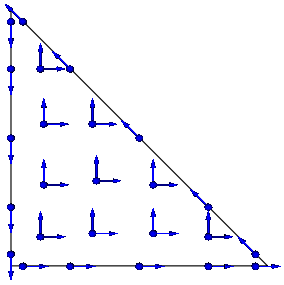


Figure 3.12: *Locations and directions of $H(curl, \hat{\kappa})$ basis functions of order $p = 5$.*

Figure 3.12 shows the locations and directions for $\mathbf{Q}^5(\hat{\kappa})$. The finite element space is defined to be

$$\mathbf{Q}^p(\Omega) = \{\boldsymbol{w} \in H(curl, \Omega) \mid \boldsymbol{w}|_\kappa = J_\kappa^{-T} \hat{\boldsymbol{w}} \circ F^{-1}, \hat{\boldsymbol{w}} \in \mathbf{Q}^p(\hat{\kappa})\} \tag{3.46}$$

where $F : \hat{K} \to K$ is an affine transformation from the reference to the physical space and J_F its Jacobian. The transformation in (3.46) ensures that the tangent component of the fields are preserved under the reference to physical mapping.

For brevity, we use the following notation

$$V^p = V^p(\Omega), \qquad \mathbf{Q}^p = \mathbf{Q}^p(\Omega). \tag{3.47}$$

The space $\mathbf{Q}^p(\Omega)$ admits the following approximation property [50, 51],

Theorem 3.3.1. *There exists a $C > 0$, independent of h such that for every $\boldsymbol{u} \in \mathbf{Q}^p \cap [H^{p+1}]^2$,*

$$\|\boldsymbol{u} - \Pi\boldsymbol{u}\|_0 \leq Ch^p\|u\|_p \tag{3.48}$$

$$\|\nabla \times \boldsymbol{u} - \Pi\boldsymbol{u}\|_0 \leq Ch^p\|u\|_p \tag{3.49}$$

where $\Pi : \mathbf{Q}^p \cap H^{p+1} \to \mathbf{Q}^p(\Omega)$ is the projection operator onto the finite element subspace.

Theorem 3.3.1 states that convergence for elements in $\mathbf{Q}^p$ have the same rate in the curl norm and in the L_2 norm.

3.3.2 Multigrid methods

The multigrid method is fully defined by its smoother and interpolation scheme. As with all $H(curl)$ discretizations, hybrid smoothing must be used and hence in this section we construct discrete gradient operators for interpolatory high-order bases. We then introduce two interpolation operators. The first is an extension of the p-coarsening scheme for hierarchical bases adapated to the interpolatory basis and the second is constructed by extending the multigrid methods that proved to be effective for high-order H^1 problems.

Construction of high-order discrete gradient

It was seen in Section 3.1.3 that smoothing of functions in the gradient space is necessary for lowest order edge multigrid. In order to smooth such functions a discrete gradient operator $\mathbb{D}_1$ was used to project A_1 to the gradient space $\mathbb{D}_1^T A \mathbb{D}_1$. For the lowest order case, the construction of $\mathbb{D}_1$ is simple as shown in Section 3.1.3. However, such an operator for high-order bases is not as easily obtained. Furthermore, because high-order basis functions vary (e.g. hierarchical or interpolatory) and in the case of the latter, the nodal locations, the realization of the discrete gradient is dependent on the basis used. Thus, it is difficult to construct such an operator using only topological information as was the case for the lowest order elements. For the lowest order elements, this was possible because H^1 degrees of freedom are located on the nodes of the mesh and $H(curl)$ degrees of freedom coincide with the edges of the mesh. Because $\mathbf{Q}^p$ is mimetic, and hence satisfies the de Rham diagram on the discrete levels, there is always an associated nodal space V^p and a discrete gradient operator that maps elements of V^p into the nullspace of the curl operator. In this section we show how to construct such an operator $\mathbb{D}_p : V^p \to \mathbf{Q}^p$ for high order bases.

The operator $\mathbb{D}_p$ expresses the gradients of V^p as linear combinations of elements in $\mathbf{Q}^p$. Because $\mathbf{Q}^p$ is mimetic, the gradients of each basis function in V^p make up the nullspace of the curl operator and hence are representable using the $\mathbf{Q}^p$ basis. Therefore, our goal is to find $\boldsymbol{\alpha} = \{\alpha_1, ..., \alpha_{|\mathbf{Q}^p|}\}$ such that

$$\nabla\phi_i = \sum_{j=1}^{|\mathbf{Q}^p|} \alpha_j \boldsymbol{w}_j, \qquad \forall i \in [1, |V^p|]. \tag{3.50}$$

Thus for each $\phi_i \in V^p$, the solution $\boldsymbol{\alpha}$ of (3.50) forms column i of $\mathbb{D}_p$. Because functions in V^p and $\mathbf{Q}^p$ have local support, the $\mathbb{D}_p$ can be constructed efficiently in a fashion similar to standard finite element assembly. Furthermore, because of the interpolatory nature of the basis, a linear system does not need to be solved on each element. In fact, the reference discrete gradient $\mathbb{D}_p\hat{\kappa}$ can be found as follows

$$\mathbb{D}_p\hat{\kappa}(i,j) = \nabla\phi_j(\boldsymbol{y}_i) \cdot \tau_i \tag{3.51}$$

where $\boldsymbol{y}_i$ and τ_i are the nodal locations and tangent directions for the basis functions in $\mathbf{Q}^p(\hat{\kappa})$ as defined in (3.44). Because $\nabla\phi$ for $\phi_i \in V^p(\kappa)$ scales in the same manner as $\boldsymbol{w} \in \mathbf{Q}^p(\kappa)$. Therefore, the coefficients in $\mathbb{D}_p$ represent the linear combinations of basis functions in $\mathbf{Q}^p$ that represents the gradient fields and need only be computed for the reference element.

The construction of the global discrete gradient can be done in a process that resembles finite element assembly. The reference discrete gradient is computed using (3.51), and then those values are inserted into the global matrix. However, differing from standard finite element assembly where values are summed, the values are inserted into the global matrix only if they have not been inserted previously. This is due to the fact that the tangent component is continuous across boundaries and the coefficients represent the representation of the gradient in $\mathbf{Q}^p$.

Each column of $\mathbb{D}_p$ is the representation of a gradient field, which leads to

$$S_p\mathbb{D}_p = \mathbf{0} \tag{3.52}$$

and hence is a basis for the nullspace of the curl-curl stiffness matrix. Once $\mathbb{D}_p$ is constructed we are able to perform hybrid smoothing on high-order problems using Algorithm 2.

Construction of interpolation operator

The second aspect in defining a multigrid method is the interpolation and restriction operators that maps between the fine and coarse grids. In Section 3.1.1 we use low-order approximations of the high-order problem

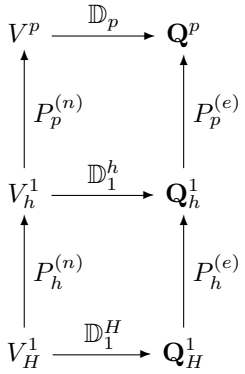


Figure 3.13: *De Rham diagram for high-order using lowest-order elements on refined mesh as coarse grid.*

is an efficient preconditioner for the H^1 case. For the H^1 case, the degrees of freedom for the high-order problem exactly match with the degrees of freedom of the low-order approximation. This is due to the fact that the Delaunay triangulation used in (3.2) uses the existing high-order degress of freedom as the vertices of the low-order mesh. However, such an extension to the $H(curl)$ case is nontrivial due to the fact that triangulations introduce new edges and hence the degrees of freedom of the high-order problem and the low-order approximation do not coincide.

Another approach is to use p-multigrid coarsening schemes where a the multigrid hierarchy is chosen to be varying order approximations to the problem. For the hierarchical basis in Section 3.2, such a coarsening scheme is natural due to the fact that the spaces are nested. Therefore, the interpolation/restriction is taking a subset of the degrees of freedom as the coarse grid. However, for an interpolatory basis, all basis functions are of the same order so such a splitting is more difficult to obtain.

In this section we explore both such methods. We show that although the degrees of freedom for the low-order approximation do not coincide with the high-order degress of freedom, it can be used as a *coarse* grid. To do this, we introduce an interpolation scheme that induces such a coarse grid. This method attempts to extend the low-order De Rham diagram of Figure 3.5 to its high-order couterpart shown in Figure 3.13. In Figure 3.13, the discrete gradient operators are obtained using the methods defined in the previous section. The edge interpolation operators $P_p^{(e)}$ are of interest and in this section we show this construction. Furthermore, in our second approach, we show how ideas from p-multigrid coarsening can be applied to the $H(curl)$ case. In this framework, the multigrid hierarchy will follow Figure 3.9 and we show how those interpolation operators can be adapted for the interpolatory case. We refer to the former as the "low-order method" and the later as the "p-multigrid method". In both methods, once the lowest-order is reached, we invoke the methods for lowest-order edge elements from Section 3.1.2. Therefore, in the following, we only

show how to obtain the high-order interpolation operators.

We first detail the low-order method. The idea of preconditiong high-order systems by low-order approximations was seen in Section 3.1.1. In the H^1 case, the points used to define the degrees of freedom for the high-order discretization was triangulated to form a low-order mesh upon which a low-order problem was discretized. The low-order discretization is an effective preconditioner for the high-order problem that can also be efficiently solved using multigrid methods. In this case, it is convenient that the number of degrees of freedom in the low-order discretization is exactly the same as that in the high-order discretization. Therefore, we can directly precondition the high-order problem using the low-order one. Since the number of degrees of freedom of the low-order approximation do not match with the that of the high-order problem, The low-order approximation cannot be used as a preconditioner. However, one can put this idea into a multigrid framework and instead use the low-order approximation as a "coarse-grid". Immediately, two problems arise: remeshing the problme so that low-order discretizations have the fidelity to accurately approximate the high-order problem and inerpolation between low-order and high-order approximations. We now address approaches to these two problems.

Concerning the first problem, it is not immediately clear how to form a low-order mesh since remeshing involves additional edges which changes the number of degrees of freedom. An obvious choice is to use the same approach as that for the H^1 case and use the nodes of the high-order $H(curl)$ basis as nodes in the new mesh. However, in doing so, the edge degrees of freedom on the old mesh now become nodes of the new mesh – where edge degrees of freedom do not lie.

To alleviate this problem, we choose to triangulate the nodes from the associated H^1 discretization of the same order. For example, Figure 3.14 shows an overlay of the high-order $H(curl)$ degrees of freedom on the low-order H^1 mesh. It is clear that on the boundary of the element, the degrees of freedom remain on

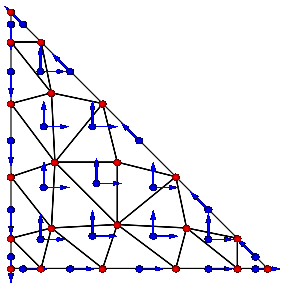


Figure 3.14: *High-order $H(curl)$ dofs (blue) on low order H^1 mesh (red).*

new edges and the number of new edges on the boundary is equal to the number of high-order degrees of freedom. However, on the interior, it is difficult to form a mesh so that the edges match with the high-order

degrees of freedom. We therefore interpolate between the high-order and low-order solutions.

To define the interpolation between the two spaces, we use a least-squares minimization principle. Thus, to define the high-order edge interpolation operator $P_p^{(e)}$, we solve for each $\boldsymbol{v}_i \in \mathbf{Q}^p(\mathcal{K})$,

$$\text{Find } \boldsymbol{u} \in \mathbf{Q}^1(\hat{\mathcal{K}}) \text{ that minimizes } \|\boldsymbol{u} - \boldsymbol{v}_i\|_0 \tag{3.53}$$

The solution to the minimization problem is given by

$$\boldsymbol{u} = M_1^{-1} M_{1,p} \boldsymbol{v} \tag{3.54}$$

where M_1 denotes the mass matrix for $\mathbf{Q}^1(\hat{\mathcal{K}})$ and $M_{1,p}$ denotes the matrix discretizing $\int_\Omega \boldsymbol{u}\cdot\boldsymbol{v}$ for $\boldsymbol{u} \in \mathbf{Q}^1(\hat{\mathcal{K}})$ and $\boldsymbol{v} \in \mathbf{Q}^p(\mathcal{K})$. An approximation to the low order mass matrix can be obtained by using a mass lumping scheme where the off diagonal entries and combined to the diagonal. Such schemes are often used to approximate mass matrices. Although the size of the low-order problem is usually larger than the size of the high-order problem, the sparsity of the low-order problem compared with the high-order problem allows the hierarchy to have low operator complexity.

We now address how the extension of the p-multigrid method used in Section 3.2 to interpolatory bases. This method is useful because the auxiliary low-order problem does not need to be rediscretized. However, because the interpolatory basis does not satisfy the nested space property (3.36), the interpolation operators are nontrivial. Any low-order basis can be written as a linear combination of high-order basis functions, the interpolation operator plays this roll.

Henceforth we consider spaces $\mathbf{Q}^q$ and $\mathbf{Q}^p$ with $q < p$. Because both spaces are interpolatory, they each satisfy the Kronecker delta property of (3.44). Our goal is to find a representation of $\mathbf{Q}^q$ in terms of the basis $\mathbf{Q}^p$. For the reference element, this representation can be found by solving the following generalized Vandermonde matrix and right hand side

$$V_{pq} = \begin{bmatrix} \boldsymbol{q}_0^p(\boldsymbol{x}_0)\cdot\tau_0^p & \boldsymbol{q}_1^p(\boldsymbol{x}_0)\cdot\tau_1^p & \cdots & \boldsymbol{q}_n^p(\boldsymbol{x}_0)\cdot\tau_n^p \\ \boldsymbol{q}_0^p(\boldsymbol{x}_1)\cdot\tau_0^p & \boldsymbol{q}_1^p(\boldsymbol{x}_1)\cdot\tau_1^p & \cdots & \boldsymbol{q}_n^p(\boldsymbol{x}_1)\cdot\tau_n^p \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{q}_0^p(\boldsymbol{x}_m)\cdot\tau_0^p & \boldsymbol{q}_1^p(\boldsymbol{x}_m)\cdot\tau_1^p & \cdots & \boldsymbol{q}_n^p(\boldsymbol{x}_m)\cdot\tau_n^p \end{bmatrix}, \quad b_j = \begin{bmatrix} \boldsymbol{q}_j^q(\boldsymbol{x}_0)\cdot\tau_0^q \\ \boldsymbol{q}_j^q(\boldsymbol{x}_1)\cdot\tau_1^q \\ \vdots \\ \boldsymbol{q}_j^q(\boldsymbol{x}_m)\cdot\tau_m^q. \end{bmatrix} \tag{3.55}$$

For the matrix V_{pq}, $\{\boldsymbol{q}_0^p, ..., \boldsymbol{q}_n^p\}$ denote the basis functions in $\mathbf{Q}^p$ and $\{\tau_0^p, ..., \tau_n^p\}$ their associated directions.

$\{\boldsymbol{x}_0, ..., \boldsymbol{x}_m\}$ denote the locations associated with the degrees of freedom for each basis function in $\mathbf{Q}^q$. For the right hand side b_j, $\boldsymbol{q}_j^q$ denotes the jth basis function in $\mathbf{Q}^q$ and $\{\tau_0^p, ..., \tau_m^q\}$ denotes the directions of those basis functions. The system $V_{pq}x = b_j$ is solved for each basis function in $\mathbf{Q}^q$.

The matrix V_{pq} is under-determined and therefore has infinitely many solutions. We therefore use the minimum norm solution. Because the Vandermonde matrix in (3.55) uses the higher order basis $\mathbf{Q}^p$, the resulting solution to $V_{pq}x = b_j$ is the jth row of the restriction operator. However, one could also choose to write the Vandermonde matrix in terms of the lower order basis and find a least-squares solution. The construction of the restriction operator is summarized in Algorithm 5.

Algorithm 5: Construct R(p, q)

Input: p, high-order basis; q, lower-order basis
return: R, restriction operator
$B = [b_0, ..., b_n]$
$R_{loc} \leftarrow \min \|R_{loc}\|_F$ such that $V_{pq}R_{loc} = B$ {minimum norm solution}
$R \leftarrow$ `assemble`(R_{loc}) {global assembly}

3.3.3 Computational study

In this section we demonstrate the effectiveness of the method presented in Section 3.3.1 through a series of numerical studies. The discrete matrices are obtained using the basis from the Intrepid package [21] of Trilinos [38]. The multigrid solver is then implemented using PyAMG [7], an algebraic multigrid package for Python. The right handside b is obtained by $Ax = b$ where x is a random vector. The initial guess is the zero vector. Unless otherwise noted, all of our tests the method is used as a preconditioner to conjugate gradient. Iterations are run until $\|b\|/\|r\| \leq 10^{-8}$.

Low-order method

We first show some properties of the Nedelec elements and their discrete matrices. We look at the approximation properties stated in Theorem 3.3.1 when we use the manufactured solution

$$\boldsymbol{u} = \begin{bmatrix} \sin(2\pi x)\sin(3\pi y) \\ \\ \sin(4\pi x)\sin(5\pi y) \end{bmatrix} \tag{3.56}$$

on the domain $\Omega = [0, 1] \times [0, 1]$. From Figure 3.15, it is clear that for each p, the solution at the same rate in L_2 and in $H(curl)$.

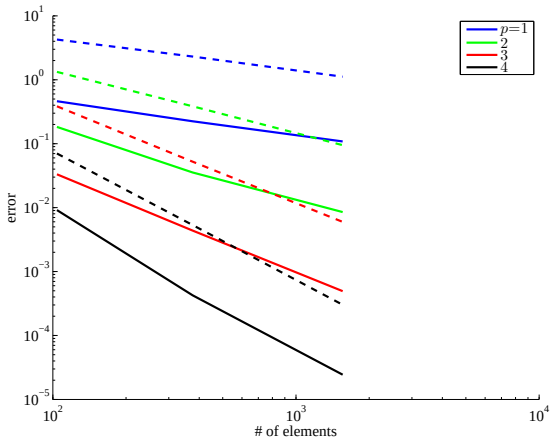


Figure 3.15: *Convergence of order $p = 1, 2, 3, 4$ triangular elements in the L_2 norm (solid) and $H(curl)$ norm (dashed).*

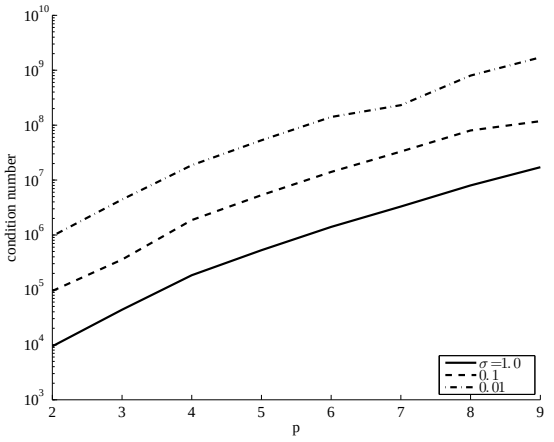


Figure 3.16: *Condition number vs. p for $\sigma = 1, 10^{-1}, 10^{-2}$.*

As expected, the condition number increases with the order of approximation p. Furthermore, The factor σ in (2.8), varies the weight of the mass matrix term. In electromagnetic applications, σ denotes the conductivity and a small σ is common in practice. However, in a relative sense, a small σ makes the curl-curl stiffness matrix term stronger and therefore more difficult to solve. Figure 3.16 shows how the condition number depends on p for $\sigma = 1, 10^{-1}, 10^{-2}$. It can be seen that changes in σ shift the condition number of the matrix. As $\sigma \to 0$, the matrix becomes more curl-curl dominated which is singular due to the large nullspace of $\nabla\times$.

We now test the performance of the method of the method vs the order of approximation on the unit square $\Omega = [0, 1] \times [0, 1]$ for meshes of varying sizes. In the edge preconditioner, we use $V(2, 2)$ cycles, where we perform two iterations of pre and post hybrid smoothing.

# elements	p	CG	SA	Edge	Size / NNZ
28	2	159	46 / 1.04	9 / 1.79	152 / 1336
	3	391	95 / 1.01	15 / 1.30	312 / 5040
	4	689	151 / 1.00	21 / 1.25	528 / 13488
	5	1043	244 / 1.00	25 / 1.12	800 / 29560
	6	1469	347 / 1.00	32 / 1.06	1128 / 56808
	7	1968	494 / 1.00	34 / 1.06	1512 / 99456
	8	2540	687 / 1.00	41 / 1.05	1952 / 16240
	9	3185	916 / 1.00	46 / 1.05	2448 / 251208
104	2	436	143 / 1.04	18 / 2.00	544 / 5456
	3	991	218 / 1.01	26 / 1.34	1128 / 20232
	4	1883	343 / 1.00	35 / 1.23	1920 / 53472
	5	3184	482 / 1.00	45 / 1.15	2920 / 116120
	6	4467	698 / 1.00	57 / 1.07	4128 / 221616
	7	6906	963 / 1.00	66 / 1.07	5544 / 385896
	8	9321	1306 / 1.00	81 / 1.07	7168 / 627392
	9	11703	1822 / 1.00	86 / 1.07	9000 / 967032
376	2	913	250/1.06	31 / 2.11	1928 / 21904
	3	1963	386 / 1.02	40 / 1.31	4020 / 79740
	4	3560	597 / 1.00	59 / 1.23	6864 / 207936
	5	5966	849 / 1.00	80 / 1.10	10460 / 447100
	6	8722	1216 / 1.00	102/ 1.06	14808 / 846864
	7	12917	1642 / 1.00	107 / 1.05	19908 / 1465884
	8	17753	2327 / 1.00	129 / 1.05	25760 / 2371840
	9	25111	3156 / 1.00	144 / 1.05	32364 / 3641436

Figure 3.17: *Iterations of CG, smoothed aggregation PCG, and edge PCG vs. order of basis* p.

p-multigrid method

For the p-multigrid methods, our numerical tests use hexahedral elements. In our first numerical test, we choose Ω to be the unit cube $[0,1]^3$. We compute the right-hand-side vector $\boldsymbol{f}$ by choosing $\boldsymbol{u}$ to be the smooth analytic solution

$$\boldsymbol{u} = \begin{bmatrix} \sin(\pi x)\sin(\pi y)\sin(\pi z) \\ \sin(\pi x)\sin(\pi y)\sin(\pi z) \\ \sin(\pi x)\sin(\pi y)\sin(\pi z) \end{bmatrix} \tag{3.57}$$

and computing $\boldsymbol{f} = \nabla \times \nabla \times \boldsymbol{u} + \sigma \boldsymbol{u}$ where σ is chosen to be a constant. We deal with discontinuous σ later in this section.

In our first numerical study we examine the scalability of our method with respect to the polynomial order p. Because the elements are of high-order, very fine meshes are generally not necessary. We test the preconditioner on meshes of sizes $h = 1/2$ to $h = 1/16$ with different order approximations and with $\sigma = 10^{-2}$. As a point of reference, we show iteration counts for conjugate gradient (CG) and smoothed aggregation AMG (SA-PCG) as a preconditioner to CG. The results are shown in Table 3.10. From the results we see that the growth in iterations is modest as p is increased and that it is comparable to the growth when h is refined. The growth in iterations with respect to mesh size is well known for this problem [20], and the edge-based preconditioner maintains this level of scalability for each p.

The number of iterations to convergence is not a complete measure of efficiency. The cost of each iteration is needed, and for this we monitor the complexity of the cycle. Operator complexity measures the amount of additional work in the multigrid cycle relative to one iteration of relaxation on the finest level. We define operator complexity as

$$\mathcal{C}_{operator} = \frac{\sum_i nnz(A_i)}{nnz(A_0)}. \tag{3.58}$$

Table 3.11 shows the operator complexities for varying p when we coarsen by $p \to p/2$ and by $p \to p-1$. We see that in the more aggresive coarsening scheme, $p \to p/2$, the operator complexity is independent of p while in the coarsening scheme where $p \to p-1$, the operator complexity is increasing with p. In either case, however, the operator complexity is very low indicating an inexpensive multilevel preconditioner.

Because we employ hybrid smoothing, our smoother performs additional work per cycle. Since the additional work is done in smoothing the gradient space, we thus measure the additional work by

$$\mathcal{C}_{hybrid} = \frac{\sum_i nnz(D_i^T A_i D_i)}{nnz(A_0)} \tag{3.59}$$

p	h	d.o.f.	CG	SA-PCG	Edge-PCG
2	1/2	300	20	34	6
	1/4	1944	129	68	13
	1/8	13872	327	141	26
	1/16	104544	654	316	51
3	1/2	1944	118	54	10
	1/4	6084	297	102	21
	1/8	45000	591	215	40
	1/16	345744	1173	462	78
4	1/2	3630	286	86	15
	1/4	13872	621	178	28
	1/8	104544	1224	342	55
5	1/2	3630	489	117	19
	1/4	26460	1140	229	36
	1/8	201720	2000+	438	71
6	1/2	6084	772	162	22
	1/4	45000	1681	337	43

Table 3.10: *Growth in iteration counts for our edge-based smoothed aggregation methods versus standard smoothed aggregation (SA) and conjugate gradient, polynomial order and mesh size are refined.*

	h	2	3	4	5	6
$p \to p/2$	1/2	1.004	1.001	1.006	1.009	1.026
	1/4	1.015	1.002	1.015	1.010	1.026
	1/8	1.025	1.003	1.022	1.009	
	1/16	1.033	1.004			
$p \to p-1$	1/2	1.004	1.142	1.259	1.385	1.517
	1/4	1.013	1.139	1.256	1.383	1.515
	1/8	1.024	1.238	1.255	1.381	
	1/16	1.033	1.137			

Table 3.11: *Operator complexity as defined in (3.58) for two different coarsening schemes.*

The additional work is summarized in Table 3.12. Because the cycling used is $V(1,1)$ cycle, and the cost of hybrid smoothing is two relaxation sweeps on the curl space and two relaxation sweeps on the gradient

space, the total cost is

$$\mathcal{C}_{total} = 2(2\mathcal{C}_{operator} + 2\mathcal{C}_{hybrid}) \tag{3.60}$$

From the results, we see that the added cost of the hybrid relaxation scheme is minimal, particular when considering the total growth in complexity. The combined results of Tables 3.10, 3.11, and 3.12, indicates that the multilevel solver exhibits only slow growth in total cost as p is increased.

	h	2	3	4	5	6
$p \to p/2$	1/2	0.144	0.138	0.141	0.137	0.138
	1/4	0.198	0.169	0.162	0.151	0.148
	1/8	0.214	0.177	0.168	0.154	
	1/16	0.217	0.179			
$p \to p-1$	1/2	0.144	0.170	0.186	0.201	0.216
	1/4	0.198	0.199	0.204	0.214	0.225
	1/8	0.214	0.205	0.208	0.216	
	1/16	0.217	0.206			

Table 3.12: *Additional work in hybrid smoother as defined in (3.59). A factor of 1.0 indicates the same cost as relaxation, thus doubling the cost of the cycle.*

We also test the case where σ is discontinuous. For this test problem, let $\Omega = [0,1]^3$ and $\Omega_1 = [0,1/3]^3$. We impose a discontinuity in σ by defining

$$\sigma = \begin{cases} 1 & \text{on } \Omega_1 \\ 10^{-3} & \text{on } \Omega \backslash \Omega_1 \end{cases} \tag{3.61}$$

As shown in Table 3.13 the proposed method performs as well as the case for constant σ; however, the performance of CG and smoothed aggregation preconditioned CG greatly deteriorates.

p	h	CG	SA-PCG	Edge-PCG
2	1/4	327	132	13
	1/8	1325	584	25
3	1/4	1012	337	20
	1/8	2000+	1177	40
4	1/4	2000+	834	28
	1/8	2000+	2000+	55
5	1/4	2000+	1320	36
	1/8	2000+	2000+	71
6	1/4	2000+	2000+	45

Table 3.13: *Iterations vs. degree of approximation for discontinuous σ.*

Chapter 4

Implementation details

In this chapter we detail the implementation of the methods developed in this dissertation. There are numerous finite element packages available, both open source and commerical. These packages excel at quick prototyping and implementation of discrete problems using standard C^0 finite elements. However, many of the elements used in this dissertation are nonstandard such as high-order, divergence-free, discontinuous, and edge bases. We therefore choose to implement the majority of our algorithms using the Trilinos framework [38]. In particular, we utilize extensively the finite element basis package Intrepid [21].

Intrepid provides the most often used basis functions such as H^1, $H(curl)$, $H(div)$, finite volume and the high-order extensions for each of these function spaces. Furthermore, Intrepid provdes tools for different cell topologies and cubature rules on each of these topologies. Basis functions are defined on the reference element and transformation functions are provided to map values to physical space. As of the time of writing this dissertation, the functionality of Intrepid is currently in the expert version where one is required to assemble the finite element matrix using the provided tools for cubature and transformations. The more comprehensive packages such as the FEniCS package [47] abstract the low-level finite element routines for a higher-level interface familiar to mathematical notation. As mentioned previously such over encompassing packages are useful when standard elements and formulations are used, but modification of such packages for nonstandard capabilities remains difficult. For this reason, we implement our methods using Intrepid due to the accessibility of the interface for low-level finite element routines. The implementation of nonstandard finite elements is not sparse in literature and may not be available in existing packages. We therefore detail such implementations in this chapter.

In Section 4.1 we describe the mesh data structure and operations necessary for finite element computations. In Section 4.2 we review basis functions for H^1, $H(curl)$, and $H(div)$ along with their operator preserving transformations. We describe their representation as polynomials and their connectivity properties. In Section 4.3 we describe assembly of local matrices to form the global matrix.

4.1 Mesh data structures

Finite element discretizations are done on meshes. In this section we detail the mesh data structure used in this dissertation. Typical mesh generators for unstructured meshes return two multidimensional arrays. The first array is a `points` array that is $N_{pts} \times \text{dim}$ where N_{pts} is the number of nodes in the mesh and dim is the dimension of the domain. The second is the element to node array which we denote `elemToNode`. `elems` is an $N_{elems} \times k$ array where N_{elems} is the number of elements in the mesh and k is the number of nodes per element.

If one were to implement linear H^1 conforming elements, then the degrees of freedom are located on the nodes and hence the above two arrays are sufficient for implementing the methods. However, in this dissertation, we deal with $H(curl)$ elements in which degrees of freedom lie on edges and also discontinuous methods in which flux terms require the knowledge of adjacent elements. In Section 4.1.1 we describe algorithms on the above two arrays to achieve additional information such as the element to edge array, the edge to node array, and the adjacent elements array.

4.1.1 Mesh operations

High-order basis functions and $H(curl)$ basis functions have degrees of freedom that lie on edges. It is therefore necessary to introduce additional data structures that hold edge information. As such, we denote `edgeToNode`, an $N_{edges} \times 2$ array that holds the end points of each edge and `elemToEdge`, an $N_{elems} \times e$ array that holds the edges for each element. Here e is the number of edges per element.

Using `elemToNode`, we are able to obtain the `edgeToNode` efficiently using the following method.

1. Create a sparse matrix $N_{elems} \times N_{pts}$ array `E2V` such that $\texttt{E2V}(i,j) = 1$ if and only if element i contains node j. `E2V` has exactly k nonzeros per row.

2. Form the matrix $\texttt{V2V} = (\texttt{E2V})^T(\texttt{E2V})$. This matrix represents the vertex to vertex connections and hence the edges.

3. Iterate through the nonzeros in the `V2V` and insert new edges into `edgeToNode`. Since the matrix is symmetric, we need only consider the upper triangular part. Thus, if $\texttt{V2V}(i,j) = 1$, then nodes there exists an edge from node i to node j. These are inserted into the `edgeToNode` array. By examining only the upper trianglular part, all edges are ordered so that $i < j$.

A consistent direction for the edges is necessary (e.g. Step 3 of the previous procedure) during the assembly procedure which we describe in Section 4.3.

In order to form `elemToEdge` we obtain the end points of each edge by indexing the columns of the `elemToNode` array using the local edge numbering. Once we have the end points of each edge, the edge index can be looked up in the `edgeToNode` array.

The last array needed is the `adjElems` array which has dimensions $N_{elems} \times e$ that marks the adjacent element across the local edge. The following procedure computes the adjacent elements.

1. Form the matrix $\texttt{E2E} = (\texttt{E2V})(\texttt{E2V}^T)$. This matrix represents an element to element connection matrix where the values in the matrix denote how many nodes the elements share.

2. If $\texttt{E2E}(i, j) = 2$ then the elements share an edge and hence are adjacent. Insert element j into adjacent elements for element i.

4.2 Basis functions

The heart of finite element methods lies in the definition of the basis functions. Infinite dimensional function spaces are discretized by finite dimensional subspaces which are spanned by a set of basis functions

$$V^p = span\{\phi_0, ..., \phi_N\}. \tag{4.1}$$

The finite element solution u^h is given as a linear combination of basis functions

$$u^h = \sum_{i=0}^{N} \alpha_i \phi_i \tag{4.2}$$

for $\alpha_i \in \mathbb{R}$.

Basis functions are typically defined and evaluated on a reference element $\hat{\mathcal{K}}$, and the evaluated values are then transformed to a physical element $\mathcal{K}$ using appropriate transformations. Because function evaluation is usually expensive, this is the most efficient way as the basis functions need only be evaluated once (at the quadrature points for the reference element) before the finite element assembly. In some cases, for example, the divergence free basis defined in Chapter 2, the basis functions are defined on the physical element.

In this section, $\hat{\nabla}, \hat{\nabla}\times$ and $\hat{\nabla}\cdot$ correspond to operators on the reference element and $\hat{\boldsymbol{x}} \in \hat{\mathcal{K}}$. Operators and variables without $\hat{}$ are assumed to be on the physical element. We denote

$$F : \hat{\mathcal{K}} \to \mathcal{K} \tag{4.3}$$

an invertible map and J_F its Jacobian matrix. For simplices such as triangles and tetrahedron, F is affine.

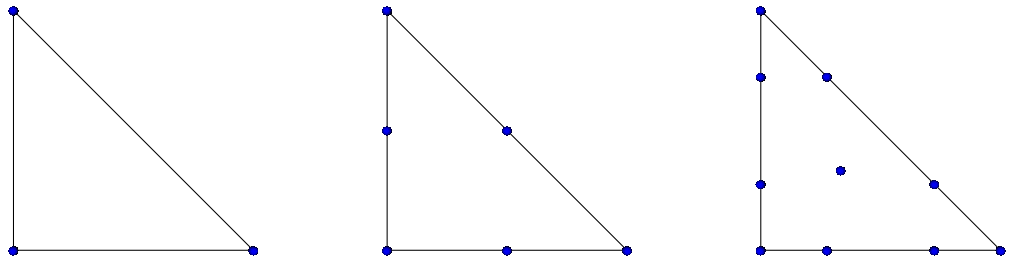


Figure 4.1: *H^1 degrees of freedom for $p = 1, 2, 3$*

4.2.1 H^1 basis

Most standard finite element schemes use H^1 conforming basis functions that are also known as nodal finite elements. These are called nodal finite elements due to the fact that the degrees of freedom in the lowest order case lie on the vertices of the mesh. As such, the basis ensures C^0 continuity across element boundaries. Figure 4.1 shows the locations of basis functions for various order polynomials. We remark that in order to obtain C^0 continuity of the element, the locations of the points cannot be chosen arbitrarily. Furthermore, the dimension of the basis is dependent on the polynomial order. For example, Figure 4.1 shows the locations of the degrees of freedom for degrees $p = 1, 2, 3$. There is always a degree of freedom at the vertices and there are $p + 1$ degrees of freedom on each edge in order to span polynomials of degree p on the edge. The rest are interior degrees of freedom to span the space of polynomials of degree p on the entire triangle. The basis functions are defined as the Lagrange interpolating basis through the specified points and therefore have the property

$$\phi_i(\boldsymbol{x}_j) = \delta_{ij}. \tag{4.4}$$

The H^1 basis is a scalar basis in which common operations are evaluation of the value of the basis and the gradient of the basis.

The values and gradients of the physical basis functions are defined via pullbacks to the reference element as

$$u(\boldsymbol{x}) = \hat{u}(\hat{\boldsymbol{x}}) \circ F^{-1}(\boldsymbol{x}) \tag{4.5}$$

and

$$\nabla u(\boldsymbol{x}) = J_F^{-T} \hat{\nabla}(\hat{\boldsymbol{x}}) \circ F^{-1}(\boldsymbol{x}) \tag{4.6}$$

respectively.

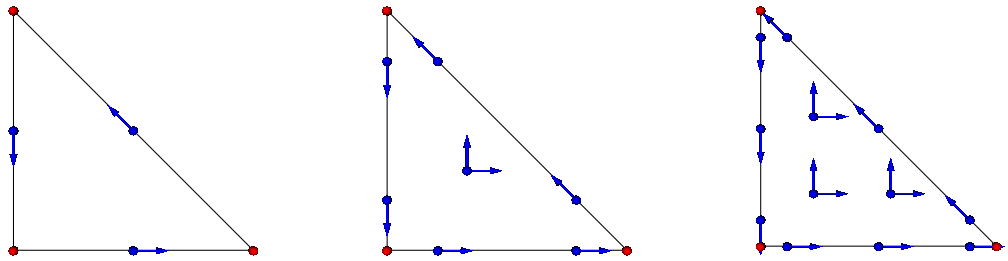


Figure 4.2: *$H(curl)$ degrees of freedom and directions for $p = 1, 2, 3$. Basis functions that lie on edges are tangent to the edge.*

4.2.2 $H(curl)$ basis

$H(curl)$ conforming basis functions are vector basis functions that have square integrable curls. As such, the basis functions are now defined by their location and a direction. $H(curl)$ bases enforce tangential continuity across element boundaries and hence specify the tangent component of the field. Figure 4.2 shows the locations and directions for various order basis functions. The basis spans polynomials of degree $p-1$ for the tangential component on the edges.

The basis has the property that

$$\boldsymbol{u}_i(\boldsymbol{x}) \cdot \tau_j = \delta_{ij} \tag{4.7}$$

so that the tangent component of the edge associated with each degree of freedom satisfies the Kronecker delta property.

The values and curls of physical basis functions are defined via pullbacks to the reference element by

$$\boldsymbol{u}(\boldsymbol{x}) = J_F^{-T} \hat{u}(\hat{\boldsymbol{x}}) \circ F^{-1}(\boldsymbol{x}) \tag{4.8}$$

and

$$\nabla \times \boldsymbol{u}(\boldsymbol{x}) = \frac{1}{|J_F|} J_F \hat{\nabla} \times \hat{\boldsymbol{u}}(\hat{\boldsymbol{x}}) \circ F^{-1}(\boldsymbol{x}) \tag{4.9}$$

in three dimensions and

$$\nabla \times \boldsymbol{u}(\boldsymbol{x}) = \frac{1}{|J_F|} \hat{\nabla} \times \hat{\boldsymbol{u}}(\hat{\boldsymbol{x}}) \circ F^{-1}(\boldsymbol{x}) \tag{4.10}$$

in two dimensions.

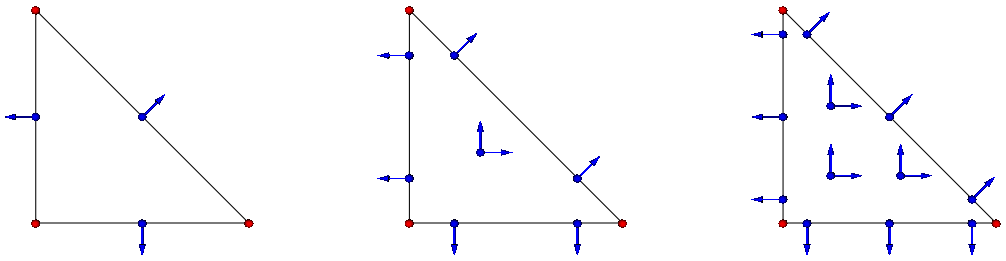


Figure 4.3: *$H(div)$ degrees of freedom and directions for $p = 1, 2, 3$. Basis functions that lie on faces are normal to the face.*

4.2.3 $H(div)$ basis

$H(div)$ conforming basis functions are vector basis functions that have square integrable divergence. $H(div)$ bases enforce normal continuity across element boundaries and hence specify the normal component of the field. Figure 4.3 shows the locations and directions for various order basis functions.

The basis has the property that

$$\boldsymbol{u}_i(\boldsymbol{x}) \cdot \boldsymbol{n}_j = \delta_{ij} \tag{4.11}$$

so that the normal component of the face associated with each degree of freedom satisfies the Kronecker delta property.

The values and divergences of physical basis functions are defined via pullbacks to the reference element by

$$\boldsymbol{u}(\boldsymbol{x}) = \frac{1}{|J_F|} J_F \hat{u}(\hat{\boldsymbol{x}}) \circ F^{-1}(\boldsymbol{x}) \tag{4.12}$$

and

$$\nabla \times \boldsymbol{u}(\boldsymbol{x}) = \frac{1}{|J_F|} \hat{\nabla} \cdot \hat{\boldsymbol{u}}(\hat{\boldsymbol{x}}) \circ F^{-1}(\boldsymbol{x}). \tag{4.13}$$

(4.12) is known as the *Piola* transform.

4.3 Finite element assembly

A major process in the finite element solution of PDEs is the assembly of the discrete matrix. Variational problems are posed as the following. Let U, V be Hilbert spaces, $a : U \times V \to \mathbb{R}$ a bilinear form, and

$f : V \to \mathbb{R}$ a linear functional. Then variational problems are posed as: *seek* $u \in U$ such that

$$a(u, v) = (f, v) \tag{4.14}$$

for all $v \in V$. The spaces U and V are known as *trial* and *test* function spaces respectively. If $U = V$ then the methods are known as Bubnov-Galerkin, otherwise they are known as Petrov-Galerkin. The discretization of $a(\cdot, \cdot)$ uses the fact that

$$U = span\{u_1, ..., u_N\} \tag{4.15}$$

and

$$V = span\{v_1, ..., v_M\}. \tag{4.16}$$

Thus, expanding (4.14) in terms of the basis for U and V along with (4.2) the variational problem becomes *seek* $\alpha_1, ..., \alpha_N$ *such that*

$$\sum_{i=1}^{N} \alpha_i a(u_i, v_j) = (f, v_j)_0 \tag{4.17}$$

for all $v_j \in V$. The bilinear form can be written as a matrix A such that

$$A_{ij} = a(u_j, v_i) \tag{4.18}$$

and the linear functional is a vector b such that

$$b_i = (f, v_i)_0. \tag{4.19}$$

We therefore solve the matrix for $Ax = b$ where $x = (\alpha_1, ..., \alpha_N)^T$. Here we observe that the *rows* of A correspond to the test function space V and the *columns* of A correspond to the trial function space U.

The assembly of a global finite element matrix is done by inserting (summing) local values into a global matrix. Weak forms of PDEs are given as sums of individual elements. For example, the variational problem for the Poisson equation is: *find* $u \in H^1$ *such that*

$$\int_\Omega \nabla u \cdot \nabla v = \int_\Omega fv \tag{4.20}$$

for all $v \in H^1$. The domain Ω is partitioned into elements so that $\Omega = \kappa_0 \cup \kappa_1 \cup ... \cup \kappa_n$. Thus, (4.20) can be written as a sum of integrals

$$\sum_{i=0}^{n} \int_{\kappa_i} \nabla u \cdot \nabla v = \sum_{i=0}^{n} \int_{\kappa_i} fv \tag{4.21}$$

The finite element problem is reduced to integrals on local elements. The local integrals are then summed to form the global matrix.

While forming the global matrix from local elemental matrices, it is crucial that on element interfaces, the correct degrees of freedom are matched and summed. When the edge has multiple degrees of freedom, they are usually numbered and ordered in the direction of the edge. Therefore, an edge direction as described in Section 4.1.1 is sufficient to ensure that local to global mapping is consistent.

In the case of vector elements, it is necessary to ensure that the direction of the local degree of freedom matches that of the global degree of freedom. As described in Section 4.1.1, all global edges are directed from lower numbering to higher numbering. Thus, one way to enforce the consistency in the direction of the basis functions is by following the global edge direction. If the local edge direction does not match the global edge direction, then it is necessary to flip the degree of freedom by multiplying by -1.

Because the basis functions, their derivatives, and their restriction to boundaries are polynomials, they can be integrated exactly using high enough order numerical quadrature rules. The Intrepid package provides quadrature rules of various orders for different element types such as lines, triangles, quadrilaterals, etc.

4.3.1 Boundary conditions

In least-squares methods boundary conditions can be set either weakly or strongly. If the boundary conditions are set weakly, then corresponding terms are added to the functional to be minimized and hence is part of the assembly process. This approach was done in many cases in Chapter 2. A weakly set boundary minimizes the error on the boundary in the functional; however, does not satisfy the boundary conditions exactly. A sufficient method to satisfy the boundary conditions exactly is by setting them strongly in which case modifications to both the discrete matrix and the right hand side need to be made.

In this disseratation we implemented homogeneous and nonhomogeneous Dirichlet boundary conditions. We have that $u = 0$ on Γ and $u = g$ on Γ for homogeneous and nonhomogeneous boundary conditions respectively. The finite element spaces for the variational problem are always closed subspaces, therefore, for nonhomogeneous problems, it is necessary to reduce the problem to the homogeneous case. We let $V_0 \subset V$ denote the closed subspace of V with vanishing trace. Let $u_{\mathcal{I}} \in V_0$ and $u_{\mathcal{D}}$ be a trace function that is nonzero on the boundary, then $u \in V$ can be written as

$$u = u_{\mathcal{I}} + u_{\mathcal{D}}. \tag{4.22}$$

Using (4.22), the weak formulation (4.14) can be viewed as

$$\begin{aligned} a(u,v) &= (f,v) && (4.23)\\ a(u_{\mathcal{I}}+u_{\mathcal{D}},v) &= (f,v) && (4.24)\\ a(u_{\mathcal{I}},v) &= (f,v)-a(u_{\mathcal{D}},v) && (4.25) \end{aligned}$$

To implement (4.25) we first obtain a matrix A and a right hand side b that discretizes the PDE as if all nodes were interior. We then form a *boundary vector* u_0 such that

$$u_0(i) = \begin{cases} 0 & \text{if d.o.f. } i \text{ located on interior} \\ g(\boldsymbol{x}_i) & \text{if d.o.f. } i \text{ located on boundary} \end{cases} \tag{4.26}$$

The right hand side vector b is updated using

$$b \leftarrow b - Au_0. \tag{4.27}$$

The boundary degrees of freedom are now treated as homogeneous Dirichlet and can therefore be removed from A by zeroing out the corresponding row and column and inserting a 1 on the diagonal. The corresponding row of the right hand side vector b is set to 0.

If x is the solution to $Ax = b$, then the final solution can be obtained by setting

$$x \leftarrow x + u_0 \tag{4.28}$$

Chapter 5

Summary and future directions

This dissertation focused on two of the main aspects in numerical PDEs, accurate discretizations and efficient solvers. In particular, we examined mass conservation in least-squares finite element methods and multigrid methods for high-order $H(curl)$ conforming finite elements. In terms of discretizations, we concentrated on improving mass conservation for the Stokes equations.

To improve mass conservation we introduced discontinuous elements into the least-squares framework. However, we showed through numerical experiments that discontinuous elements alone do not improve mass conservation, and for some test domains, dramatically reduces the mass conservation. To alleviate this problem, we introduced the stream function-vorticity-pressure (SVP) formulation which was shown to almost completely eliminate mass loss. We improved upon the SVP formulation by using a divergence-free basis for the velocity. However, a straight forward formulation with the divergence-free basis is insufficient to reduce mass loss to similar levels as the SVP formulation. Motivated by the SVP formulation, we augmented the formulation by imposing continuity on an implicit stream function and arrived at the discontinuous velocity-vorticity-pressure (dV-VP) formulation. The addition of this term was shown to improve mass conservation in all test problems. To complement the new formulations, we analyzed the relative scaling of the matrix blocks and introduced a simple diagonal preconditioner that balanced the relative scaling and thereby reduced the growth in condition number down to levels commensurate with standard Galerkin formulations. We apply the discontinuous methods for the Stokes equations to the more difficult Navier-Stokes equations. Here we used Newton's method to solve the nonlinear system and with the correct choice of weights, we were able to obtain results that matched those of well known methods.

In this dissertation, we introduced and extensively studied discontinuous least-squares finite element methods for the Stokes equations. Our approach is systematic and flexible in that we break the finite element space into elementwise spaces and then impose local constraints on each element. A weak sense of continuity is recovered by including jump terms in the functional. Although our focus is mass conservation for the Stokes and Navier-Stokes equations, one can apply a similar process for other problems.

This dissertation showed that discontinuous finite elements with divergence-free bases leads to improved

mass conservation for the Stokes equations. We began with the well-posed formulation for standard H^1 spaces. One possible future direction for this research is the mathematical foundations for the well-posedness of the discontinuous least-squares formulations. For discontinuous Galerkin methods, proofs for ellipticity and error estimates are well-developed for a number of problems however, they utilize integration by parts to obtain the flux terms that result in the bilinear form. On the other hand, proofs for least-squares finite elements start show that the functional is norm equivalent to the given Hilbert space. When dealing with norms, integration by parts is not a natural technique and hence methods used for discontinuous Galerkin type proofs do not carry over immediately. Therefore, future directions include proofs that discontinuous least-squares functionals, such as those given in Chapter 2, are norm equivalent on the broken function spaces.

The second main topic of this dissertation is efficient multigrid methods for matrices that result from high-order $H(curl)$ conforming discretizations. In Chapter 3 we introduced multigrid methods that efficiently solve systems discretized by both hierarchical and interpolatory type basis functions. For the hierarchical type basis functions, we utilized p-type coarsening schemes along with hybrid smoothing at each level. In order to smooth the gradient space, a high-order discrete gradient operator was introduced. For this basis, the discrete gradient operator is simple since gradient functions are used as basis functions in the high-order $H(curl)$ space. We adapted both the discrete gradient and p-type coarsening schemes to the interpolatory basis. In this case, the discrete gradient operator is nontrival and we described an algorithm to construct it. Furthermore, we adapted methods for high-order H^1 finite elements to the $H(curl)$ case. Because the degrees of freedom for the low-order approximation do not match with the high-order degrees of freedom, as was the case in H^1, we chose to use the low-order approximation as a coarse grid. The interpolation operator between the high-order and low-order problems was obtained using a minimization approach. All proposed methods converged in a small number of iterations and the cost measured in terms of operator complexity was low.

Future directions include multigrid for high-order $H(div)$ finite elements using the methods described in this dissertation. It will be necessary to utilize hybrid smoother wherein the kernel of the divergence operator is smoothed. In this case, a discrete-curl matrix needs to be defined. For the lowest-order, such operators have been developed in the field of discrete exterior calculus. Their high-order counterparts can be obtained using similar algorithms developed in this dissertation for high-ordere discrete gradients operators.

References

[1] D. Arnold, R. Falk, and R. Winther. Multigrid in $H(\mathrm{div})$ and $H(\mathrm{curl})$. *Numerische Mathematik*, 85(2), 2000.

[2] D. N. Arnold, R. S. Falk, and R. Winther. Finite element exterior calculus, homological techniques, and applications. *Acta Numerica*, 15:1–155, 2006.

[3] T. Austin, T. Manteuffel, and S. McCormick. A robust approach to minimizing *H(div)* dominated functionals in an *H1*-conforming finite element space. *J. Numer. Lin. Alg. Appl.*, 11:115–140, 2004.

[4] A. Aziz, R. Kellogg, and A. Stephens. Least-squares methods for elliptic systems. *Math. Comp.*, 44 (169):53–70, 1985.

[5] G. A. Baker, W. N. Jureidini, and O. A. Karakashian. Piecewise solenoidal vector fields and the stokes problem. *SIAM Journal on Numerical Analysis*, 27(6):1466–1485, 1990.

[6] N. Bell and L. Olson. Algebraic multigrid for k-form Laplacians. *Numer. Linear Algebra Appl.*, 15(2): 165–185, 2008.

[7] W. N. Bell, L. N. Olson, and J. Schroder. PyAMG: Algebraic multigrid solvers in Python, 2008. URL `http://www.pyamg.org`. Version 1.0.

[8] R. Bergström and M. Larson. Discontinuous least-squares finite element method for the div-curl problem. *Numerische Mathematik*, 101:601–617, 2005.

[9] C. Bernardi, G. Coppoletta, V. Girault, and Y. Maday. Spectral methods for the Stokes problem in stream-function formulation. *Computer Methods in Applied Mechanics and Engineering*, 80(1-3):229 – 236, 1990.

[10] P. Bochev. *Least-squares finite element methods for the Stokes and Navier-Stokes equations*. PhD thesis, Virginia Polytechnic Institute and State University, Blacksburg, 1994.

[11] P. Bochev. Negative norm least-squares methods for the velocity-vorticity-pressure Navier-Stokes equations. *Numerical Methods for PDEs*, 15:237–256, 1999.

[12] P. Bochev and D. Day. Analysis and computation of a least-squares method for consistent mesh tying. *J. Comp. Appl. Math*, 218:21–33, 2008.

[13] P. Bochev and M. Gunzburger. Analysis of least-squares finite element methods for the stokes equations. *Math Comp.*, 64:479–506, 1994.

[14] P. Bochev and M. Gunzburger. Analysis of least-squares finite element methods for the Stokes equations. *Math. Comp.*, 63:479–506, 1994.

[15] P. Bochev and M. Gunzburger. Least-squares for the velocity-pressure-stress formulation of the Stokes equations. *Comput. Meth. Appl. Mech. Eng.*, 126(3-4):267–287, 1995.

[16] P. Bochev and M. Gunzburger. Analysis of least-squares finite element methods for the navier-stokes equations. *SIAM J. Numer. Anal.*, 34(5):1817–1844, 1997.

[17] P. Bochev and M. Gunzburger. Finite element methods of least-squares type. *SIAM Review*, 40(4): 789–837, 1998.

[18] P. Bochev and M. Gunzburger. *Least-Squares Finite Element Methods*. Springer, 2009.

[19] P. Bochev and M. Gunzburger. A locally conservative mimetic least-squares finite element method for the Stokes equations. In I. Lirkov, S. Margenov, and J. Wasniewski, editors, *Proceedings of LSSC 2009*, volume 5910 of *Springer Lecture Notes in Computer Science*, pages 637–644., 2009.

[20] P. Bochev, C. Garasi, J. Hu, A. Robinson, and R. Tuminaro. An improved algebraic multigrid method for solving Maxwell's equations. *SIAM J. Sci. Comput.*, 25(2):623–642, 2003.

[21] P. Bochev, D. Ridzal, and K. Peterson. Intrepid: Interoperable Tools For Compatible Discretizations. http://trilinos.sandia.gov/packages/intrepid/, 2010.

[22] P. Bochev, J. Lai, and L. Olson. A locally conservative, discontinuous least-squares finite element method for the stokes equations. *Int. J. Numer. Method. Fluids*, 68:782–804, 2012.

[23] P. Bochev, J. Lai, and L. Olson. A non-conforming least-squares finite element method for the velocity-vorticity-pressure stokes equations. *Submitted*, 2012.

[24] J. Bramble and J. Pasciak. Least-squares methods for Stokes equations based on a discrete minus one inner product. *J. Comp. Appl. Math.*, 74:155–173, 1996.

[25] S. Brenner and R. Scott. *The Mathematical Theory of Finite Element Methods*. Number 15 in Texts in Applied Mathematics. Springer Verlag, New York, 2002.

[26] Z. Cai and B. Shin. The discrete first-order system least squares: the second-order elliptic boundary value problem. *SIAM J. Numer. Anal.*, pages 307–318, 2002.

[27] Y. Cao and M. Gunzburger. Least-squares finite element approximations to solutions of interface problems. *SIAM J. Numer. Anal.*, 35(1):393–405, 1998.

[28] C. Chang and J. Nelson. Least-squares finite element method for the Stokes problem with zero residual of mass conservation. *SIAM J. Numer. Anal.*, 34:480–489, 1997.

[29] B. Cockburn, G. Kanschat, and D. Schötzau. A locally conservative ldg method for the incompressible navier-stokes equations. *Math. Comp*, 74(251):1067–1095, 2004.

[30] B. Cockburn, F. Li, and C.-W. Shu. Locally divergence-free discontinuous galerkin methods for the maxwell equations. *Journal of Computational Physics*, 194(2):588 – 610, 2004. URL `http://www.sciencedirect.com/science/article/pii/S0021999103004960`.

[31] B. Cockburn, G. Kanschat, and D. Schötzau. A note on discontinuous galerkin divergence-free solutions of the navier-stokes equations. *J. Sci. Comput.*, 31(1-2):61–73, 2007.

[32] E. Dean, R. Glowinski, and O. Pironneau. Iterative solution of the stream function-vorticity formulation of the stokes problem, applications to the numerical simulation of incompressible viscous flow. *Computer Methods in Applied Mechanics and Engineering*, 87(2-3):117 – 155, 1991.

[33] J. Deang and M. Gunzburger. Issues related to least-squares finite element methods for the Stokes problem. *SIAM J. Numer. Anal.*, 35:878–906, 1998.

[34] F. Dubois, M. Salaün, and S. Salmon. Vorticity-velocity-pressure and stream function-vorticity formulations for the stokes problem. *Journal de Mathèmatiques Pures et Appliquès*, 82(11):1395 – 1451, 2003.

[35] A. Ern and J.-L. Guermond. *Theory and Practice of Finite Elements*. Number 159 in Applied Mathematical Sciences. Springer Verlag, New York, 2004.

[36] U. Ghia, K. Ghia, and C. Shin. High-re solutions to incompressible flow using the Navier-Stokes equations and a multigrid method. *J. Comput. Phys.*, 48:387–411, 1982.

[37] V. Girault and P. Raviart. *Finite Element Methods for Navier-Stokes Equations*. Springer, Berlin, 1986.

[38] M. Heroux, R. Bartlett, V. Howle, R. Hoekstra, J. Hu, T. Kolda, R. Lehoucq, K. Long, R. Pawlowski, E. Phipps, A. Salinger, H. Thornquist, R. Tuminaro, J. Willenbring, and A. Williams. An Overview of Trilinos. Technical Report SAND2003-2927, Sandia National Laboratories, 2003.

[39] J. Heys, T. Manteuffel, S. McCormick, and L. Olson. Algebraic multigrid for higher-order finite elements. *J. Comput. Phys.*, 204(2), 2005.

[40] J. Heys, E. Lee, T. Manteuffel, S. McCormick, and J. Ruge. Enhanced mass conservation in least-squares methods for Navier-Stokes equations. *SIAM J. Sci. Comput.*, 31:2303–2321, 2009.

[41] R. Hiptmair. Multigrid method for Maxwell's equations. *SIAM J. Numer Anal.*, 36(1):204–225, 1998.

[42] J. Hu, R. Tuminaro, P. Bochev, C. Garasi, and A. Robinson. Toward an h-independent algebraic multigrid method for Maxwell's equations. *SIAM J. Sci. Comput.*, 27(5):1669–1688, 2006.

[43] B.-N. Jiang. *The Least-Squares Finite Element Method. Theory and Applications in Computational Fluid Dynamics and Electromagnetics.* Springer-Verlag, New York, 1998.

[44] O. Karakashian and T. Katsaounis. Numerical simulation of incompressible fluid flow using locally solenoidal elements. *Computers & Mathematics with Applications*, 51(9-10):1551 – 1570, 2006. URL `http://www.sciencedirect.com/science/article/pii/S0898122106001015`.

[45] J. Lai and L. Olson. Algebraic multigrid for high-order hierarchical h(curl) finite elements. *SIAM J. Sci. Comput.*, 33(5):2888–2902, 2011.

[46] A. Logg and G. Wells. DOLFIN: Automated finite element computing. *ACM Transactions on Mathematical Software*, 37(20), 2010.

[47] A. Logg, K. Mardal, and G. Wells. *Automated Solution of Differential Equations by the Finite Element Method*. Springer, 2012.

[48] T. Manteuffel, K. Ressel, and G. Starke. A boundary functional for the least-squares finite element solution of neutron transport problems. *SIAM J. Numer. Anal.*, 37:556–586, 2000.

[49] M.Crouzeix and P.Raviart. Conforming and nonconforming finite element methods for solving the stationary Stokes equations. *RAIRO Model. Math. Anal. Numer.*, 3:33–76, 1973.

[50] J. C. Nédélec. Mixed finite elements in $\mathbf{R}^3$. *Numerische Mathematik*, 35:315–341, 1980.

[51] J. C. Nédélec. A new family of finite element methods in $\mathbf{R}^3$. *Numerische Mathematik*, 50:57–81, 1986.

[52] L. Olson. Algebraic multigrid preconditioning of high-order spectral elements for elliptic problems on a simplicial mesh. *SIAM J. Sci. Comput.*, 29(5), 2007.

[53] S. Orszag. Spectral methods for problems in complex geometries. *J. Comput. Phys.*, 37:70–92, 1980.

[54] M. M. Proot and M. I. Gerritsma. Analysis of a discontinuous least squares spectral element method. *J. Sci. Comput.*, 17(1-4):297–306, 2002. ISSN 0885-7474. doi: http://dx.doi.org/10.1023/A:1015173203136.

[55] P. A. Raviart and J. M. Thomas. A mixed finite element method for second order elliptic problems. In Galligani and E. Magenes, editors, *Mathematical Aspects of the Finite Element Method, I*, number 606 in Lecture Notes in Math. Springer-Verlag, New York, 1977.

[56] S. Reitzinger and J. Schöberl. An algebraic multigrid method for finite element discretizations with edge elements. *Numer. Linear Algebra Appl.*, 9(3):223–238, 2002.

[57] W. Rheinboldt. Solution fields of nonlinear equations and continuation methods. *SIAM J. Numer. Anal.*, 17:221–246, 1980.

[58] Y. Saad. *Iterative methods for sparse linear systems*. SIAM, second edition, 2003.

[59] J. Schöberl and S. Zaglmayr. High order Nédélec elements with local complete sequence property. *Int. J. Comput. Math. Electr. and Electron. Eng.*, 24(2):374–384, 2005.

[60] P. Solin, K. Segeth, and I. Dolezel. *Higher-Order Finite Element Methods*. Chapman & Hall/CRC, 2004.

[61] M. Taylor, B. Wingate, and R. Vincent. An algorithm for computing Fekete points in the triangle. *SIAM J. Numer. Anal*, 38, 2000.

[62] P. Vanek, J. Mandel, and M. Brezina. Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems. *Computing*, 56(3), 1996.

[63] T. Warburton. An explicit construction for interpolation nodes on the simplex. *Journal of Engineering Mathematics*, 56:247–262, 2006.

[64] J. Webb. Hierarchal vector basis functions of arbitrary order for triangular and tetrahedral finite elements. *IEEE Trans. Antennas and Propagation*, 47(8):1244–1253, 1999.

[65] S. Zaglmayr. *High Order Finite Element Methods for Electromagnetic Field Computation*. PhD thesis, Johannes Kepler Universität Linz, Austria, 2006.